Measuring Respiratory Frequency using Optronics and Computer Vision

Per Antonsson and Jesper Johansson
Master of Science Thesis in Electrical Engineering

Measuring Respiratory Frequency using Optronics and Computer Vision

Per Antonsson and Jesper Johansson

LiTH-ISY-EX–21/5376–SE

Supervisor: Daniel Bossér
ISY, Linköpings universitet
Astrid Lundmark
Saab Dynamics AB

Examiner: Fredrik Gustafsson
ISY, Linköpings universitet

Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Copyright © 2021 Per Antonsson and Jesper Johansson
Abstract

This thesis investigates the development and use of software to measure respiratory frequency on cows using optronics and computer vision. It examines mainly two different strategies of image and signal processing and their performances for different input qualities. The effect of heat stress on dairy cows and the high transmission risk of pneumonia for calves make the investigation done during this thesis highly relevant since they both have the same symptom; increased respiratory frequency. The data set used in this thesis was of recorded dairy cows in different environments and from varying angles. Recordings, where the authors could determine a true breathing frequency by monitoring body movements, were accepted to the data set and used to test and develop the algorithms. One method developed in this thesis estimated the breathing rate in the frequency domain by Fast Fourier Transform and was named "N-point Fast Fourier Transform." The other method was called "Breathing Movement Zero-Crossing Counting." It estimated a signal in the time domain, whose fundamental frequency was determined by a zero-crossing algorithm as the breathing frequency. The result showed that both the developed algorithm successfully estimated a breathing frequency with a reasonable error margin for most of the data set. The zero-crossing algorithm showed the most consistent result with an error margin lower than 0.92 breaths per minute (BPM) for twelve of thirteen recordings. However, it is limited to recordings where the camera is placed above the cow. The N-point FFT algorithm estimated the breathing frequency with error margins between 0.44 and 5.20 BPM for the same recordings as the zero-crossing algorithm. This method is not limited to a specific camera angle but requires the cow to be relatively stationary to get accurate results. Therefore, it could be evaluated with the remaining three recordings of the data set. The error margins for these recordings were measured between 1.92 and 10.88 BPM. Both methods had execution time acceptable for implementation in real-time. It was, however, too incomplete a data set to determine any performance with recordings from different optronic devices.
Acknowledgments

First of all, we would like to thank our supervisor at SAAB Dynamics AB, Astrid Lundmark. Her enthusiasm and expertise have made a significant impact on the progression of this thesis. Secondly, we would like to thank our supervisor at LiU, Daniel Bossér, and our examiner Fredrik Gustafsson for providing us with excellent feedback and guidelines. Finally we would like to thank SAAB Dynamics AB for supplying great resources for the development of this work.

Linköping, June 2021
Per Antonsson and Jesper Johansson
# Contents

1 Introduction
  1.1 Background ................................................. 1
  1.2 Aim .................................................. 2
  1.3 Research question ................................. 2
  1.4 Delimitations ........................................... 2
  1.5 Proposed Method ................................. 3
  1.6 Thesis Outline ................................. 3

2 Theory
  2.1 Respiratory System ........................................ 5
  2.2 Gaussian Filter ........................................ 6
  2.3 Motion Vectors ......................................... 7
    2.3.1 Horn-Schunk Method ................................ 9
    2.3.2 Lucas-Kanade Method ............................. 10
    2.3.3 Farnebäck’s Method ................................ 10
  2.4 Tukey Windowing ....................................... 12
  2.5 Point-Mass Filter .................................... 13

3 Method
  3.1 Data Collection & Validation ......................... 15
    3.1.1 Initial Data Collection .......................... 15
    3.1.2 Data Validation ................................ 16
    3.1.3 Secondary Data Collection ....................... 17
  3.2 Breathing Movement Zero-Crossing Counting .......... 18
    3.2.1 ROI Tracking .................................. 18
    3.2.2 Optical Flow with Lucas-Kanade algorithm ........ 19
    3.2.3 Extracting Breathing Movements .................. 21
    3.2.4 Filter Parameter Estimation ..................... 23
    3.2.5 Signal Estimation ................................ 25
    3.2.6 Zero-crossing Algorithm ....................... 25
  3.3 Frequency Estimation with N-Point Fast Fourier Transform .... 26
    3.3.1 Pre-processing .................................. 27
    3.3.2 N-Point Fast Fourier Transform ............... 27
3.3.3 Neighborhood Weighing ........................................ 31
3.3.4 Point-Mass Inspired Filter .................................... 31

4 Results ........................................................................ 35
  4.1 N-Point Fast Fourier Transform ................................. 35
    4.1.1 Motion vectors result ....................................... 35
    4.1.2 Frequency result ........................................... 45
  4.2 Breathing Movement Zero-Crossing Counting ............... 54
    4.2.1 Motion Vectors with Varying Parameters............... 54
    4.2.2 Estimated Frequency with Varying Image Resolution . 58
    4.2.3 Chosen Parameter Configuration .......................... 65
  4.3 Comparing Estimated BPM for the N-point FFT Method and
      the Zero-crossing Method .................................... 66

5 Discussion .................................................................... 67
  5.1 Motion vectors for N-point FFT ................................. 67
  5.2 Frequency Estimation with N-point FFT ...................... 68
  5.3 Optical Flow for Breathing Movement Zero-Crossing Counting
      ................................................................. 69
  5.4 Frequency Estimation with Breathing Movement Zero-Crossing Counting
      ................................................................. 70
  5.5 General Discussion ............................................... 71

6 Conclusions .................................................................. 73
  6.1 Research Questions ............................................... 73
  6.2 Future Work ........................................................ 74

A Used Procedures ......................................................... 79
  A.1 Zero Padding ....................................................... 79
  A.2 Fast Fourier Transform .......................................... 79

Bibliography ................................................................... 81
1

Introduction

1.1 Background

Respiratory rate is a vital sign for humans along with most mammals. In both human and animal healthcare in Sweden, the standard procedure to measure the rate is by manually monitoring the patient’s breathing cycles [1]. Some classical devices are available to record chest movements, such as the pneumograph and the respirometer [2]. Some new automatic monitoring solutions are in development where the subject wears sensors such as electrocardiogram and photoplethysmogram, but no modern techniques with the primary use to record the respiratory rate has been found in the research for this thesis.

Digital signal processing has been successfully applied to human and animal healthcare for decades, and the precision and reliability of the techniques are mostly undisputed. In addition, image processing techniques are widely used in personal use, surveillance, identification, and many more areas and have also become more and more trustworthy in recent years.

Is there any need to measure the respiratory rate of animals like cows? It is a vital sign, which increases if the cow is carrying a respiratory disease or suffering from heat stress [3]. Heat stress is a known problem for dairy cows with the potential consequence of both reduced quality and quantity of the produced milk. Respiratory diseases, like pneumonia, can also affect the milk production for a cow’s lifetime span if it is infected as a calf. While sick in the process of growing, the animal’s size is also affected in a negative way. Diseases also generate chain effects on cows and calves since the sick animal feels febrile and depressed, thus increasing the risk of capturing other diseases [4].

Respiratory rate measurements are often collected manually by a person who
counts the breaths by either watching the movement of the abdominal or the nostrils. This method is labor-intensive, leading to a risk that a cow or calf is feeling sick or overheated without being observed. There is no available information about how often these measurements are done. An autonomous solution could help cover the respiratory rate daily and be a cheap, easy, and trustful way to help milk farmers save countless liters of milk per year and consequently increase income and lower labor.

1.2 Aim

The goal of this Master Thesis is to design a methodology to autonomous measure the respiratory rate using optronics and computer vision, which consequently could help prevent diseases from being spread and to detect heat stress amongst dairy cows and calves.

1.3 Research question

While this thesis achieves the aim, the following questions will be answered:

- Can the respiratory frequency of cows be measured by computer vision and signal processing, and if so, how well does the measured frequency match the true respiratory frequency?

- Are there any performance deviations when using different types of image and signal processing algorithms, and if so, can the difference be quantified?

- Are there any performance deviations when using different types of optronic devices?

- How does the resolution of the input images correspond to the signal output and performance of the measured respiratory frequency?

1.4 Delimitations

There were some delimitations during the creation and implementation of the autonomous solution. In the problem-solving phase of the project, some solutions were discarded because of time consumption although the solution could yield better results. Apart from the deadline to finish a working demonstrator in time, there were also limitations in the data collected to implement and test the system. Since the data were collected with three specific different cameras, there can be no reassurance that the system works with other optronics. Other delimitations are that the subject is alone in the frame and stand reasonably still. Recordings, where the authors cannot determine the true respiratory rate, were not used in the data set, and there were no performance requirements on such recordings. All recordings were delimited to at least 10 frames per second.
1.5 Proposed Method

The proposed method to reach the aim presented in Section 1.2 and to answer the four research questions from Section 1.3 was to approach the problem from two different angles with a common ground. One method where the breathing rate is estimated in the frequency domain, and one method strictly in the time domain where the breathing signal itself is estimated. Since respiratory rate until now have been estimated with visual monitoring of body movements, the common ground of the two methods was determined to be optical flow. Optical flow, or motion vectors, is well documented procedure to estimate movements of an object in an image between two frames. If one of the two proposed methods were to be considered impractical or not solvable, it were to be discarded after analyzed for subsystems useful for the other method.

1.6 Thesis Outline

The relevant theory for this thesis is described in Chapter 2 followed by how the data was collected and validated together with detailed descriptions of the two algorithms methodology in Chapter 3. Chapter 4 presents how the algorithms perform with the recordings from the data set and the result from varying configurations, while Chapter 5 provides discussions regarding the methods and the acquired results. In the final chapter of this thesis, Chapter 6, the drawn conclusions are presented with answers to the research questions from Section 1.3 together with some possible future work.

Per Antonsson was in charge of the Breathing Movement Zero-Crossing Counting algorithm described in Section 3.2 while Jesper Johansson implemented the N-Point Fast Fourier Transform algorithm described in Section 3.3.
This chapter includes the theory and related work necessary to understand both the problem and method used to answer the proposed questions. The theoretic background information on the math and algorithms of the thesis will be covered together with related work where they previously have been used.

2.1 Respiratory System

To be able to estimate an animal’s respiratory frequency, some basic knowledge about the animals respiratory system is needed. The breathing of a cow can be divided in to two types: abdominal and costal breathing. The abdominal breathing is characterized by visible abdomen movements, and the costal breathing is characterized by pronounced movements of the ribs. The respiratory frequency for a cow is subject to [5]:

1. body size
2. age
3. exertion
4. environmental temperature
5. pregnancy
6. degree of filling of digestive tract
7. state of health

among other factors not listed. When analyzing one individual’s breathing rate these factors should be taken to account. The respiratory frequency is, however,
a good indicator of the health status of the animal, where the frequency increases when the cow is stressed or during a disease [3] [4]. The normal respiratory frequencies for a healthy cow while standing respectively while in sternal recumbence (subject lying down but not sleeping) are listed in Table 2.1, based on manual means done on 11 diary cows. However, a cow suffering from heat stress could reach a respiratory rate as high as 88.8 BPM [6].

Table 2.1: Respiratory Frequency for Diary Cows Under Two Different Conditions.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Range</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standing (at rest)</td>
<td>26-35</td>
<td>29</td>
</tr>
<tr>
<td>Sternal recumbency</td>
<td>24-50</td>
<td>35</td>
</tr>
</tbody>
</table>

2.2 Gaussian Filter

The data set used to implement and test the two methods of measuring respiratory rate is covered in the next chapter. But knowing it is from a barn with animals of different appearances and sizes, recorded with cameras of varying performance, pre-processing the images can be helpful. In addition to scaling and sub-sampling, applying a Gaussian filter is commonly used in image processing. A 2D Gaussian filter is a useful technique to reduce image noise and reduce details by smoothing or "blurring" the image. The smoothing filter is effectively used as a low-pass filter in both the spatial and the frequency domain. As the name indicates, the visual effect is a blurry image, but it also has an effect on reducing the image’s high-frequency components as a low pass filter. Applying a Gaussian filter means convolving a digital image, \( \omega(x, y) \), with a 2D Gaussian function established by the Gaussian distribution:

\[
G(x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{2.1}
\]

The Gaussian distribution is applied using a supporting window, \( W \), of pixels from the input image to generate a pixel value to the blurred output image. The size of the supporting window is \( (m \times n) \). This is shown in Figure 2.1 and mathematically by:

\[
\omega(x, y) \ast G(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s, t) f(x-s, y-t) \tag{2.2}
\]

where \( a = (m-1)/2 \) and \( b = (n-1)/2 \) [7].
2.3 Motion Vectors

As mentioned in Section 1.5, the common ground of the two methods to measure respiratory rate is motion vectors. Motion vector fields are often used to examine motion of subjects in a digital video. By computing the magnitude and velocity of pixels in two successive frames, a vector field is formed representing the movements, exemplified in Figure 2.2. Motion vectors are frequently used in studies with the objective to detect motion [8], [9], [10], [11].

Figure 2.1: A window filter figure showing the input window, W, (shaded) that is used for computing the corresponding pixel value for the output image.
The optical flow between two frames is apparent visual motion, which in a digital image can correspond to both the movement of a certain pixel intensity representing an object or part of an object, as well as change of pixel brightness [8]. The calculation of the movement is partly a problem of minimization, as in [9]. The minimization typically uses the luminance values of two successive frames and the evaluation uses nearby motion vectors to weigh the adjacent motion vector. Alternatively the minimization can be replaced with a block-matching algorithm as in [12]. Block-matching simply specifies a block, its position and all the pixel intensities in that area of size $(N \times N)$ from one frame, and search for the same block in the next frame.

Since every pixel in two frames is taken into account for each vector field, the choice of method to compute it can be very important depending on the image feed. Some methods, like Lucas-Kanade, only computes the vectors of a predetermined set of points in the image, usually corresponding to a corner or other area of interest. This type of method is useful when the object or objects of importance can be somewhat defined in advance. It can also give more detailed information in a short amount of time since there are fewer pixels to calculate. If using the Lucas-Kanade method to calculate the motion vector field, an advanced solution to the tracking/identification problem is required to find the thoracoabdominal movement. This is explained with more detail in 2.3.2. Other methods
use faster but not as accurate calculations to compute the difference of every pixel in the two successive images, and one of the most popular is the Horn-Schunk algorithm, presented in 2.3.1. The key assumption in the Horn-Schunk algorithm is that the digital image intensity can be assumed to be equal between two successive frames. This assumption usually worked if the frame rate is high enough. This assumption and technique were used before the turn of the millennium, so the presumption that most modern digital cameras have high enough frame rate is not too farfetched. A motion vector is denoted as \((u, v)\) in (2.3) and is shown in Figure 2.3.

\[
(u, v) = \left( \frac{dx}{dt}, \frac{dy}{dt} \right)
\]  

(2.3)

Figure 2.3: The optical flow \((u, v)\) for a pixel at \((x, y)\) in time \(t\)

Another method to estimate motion vectors is the Farnebäck algorithm, explained in 2.3.3, which is a method more designed to handle non-temporally consistent sequences.

### 2.3.1 Horn-Schunck Method

The Horn Schunck algorithm [8] uses calculus of variations [13] for the computation of an optical flow field. Like most methods for determining optical flow, the principal idea is to solve a minimization problem. In the Horn-Schunck method, the goal is to find the 2D-motion vector in the xy-plane with the minimum amount of energy

\[
E = E_d + \alpha E_r,
\]

(2.4)

where \(E_d(u, v)\), the data term, is the square sum of the The Optical Flow Constraint Equation (OFCE)

\[
I_x u + I_y v + I_t = 0
\]

(2.5)

i.e. \(E_d = (I_x u + I_y v + I_t)^2\). \(I_x, I_y\) and \(I_t\) denotes the intensity with respect to \(x, y\) and \(t\) respectively. The intensity is rewritten to match a motion vector, and
\( \alpha \) is a positive scalar to be adjusted to find the optimal value while \( E_r(u, v) \) is a regularization term. The minimization can thereby be formulated as

\[
(u^*, v^*) = \arg \min_{(u, v)} \int \int E(u, v) \, dx \, dy.
\]

(2.6)

The Horn Schunk technique, explained more in detail by S.S Mokri in [8], then iterate \((u^*, v^*)\) over neighboring pixels until they have almost the same velocity, resulting in \((u^{*,k}, v^{*,k})\) which is the velocity of the pixel after \(k\) iterations.

### 2.3.2 Lucas-Kanade Method

As mentioned in Section 2.3, the Lucas-Kanade algorithm calculates a sparse optical flow and is therefore considered a fast yet accurate method for motion detection. The algorithm works under the assumption that the intensity at a pixel and its local neighborhood is relatively constant between two consecutive frames [14]. Extension of equation (2.5) from Section 2.3.1 results in a system of equations that must satisfy the motion vector \((u, v)\)

\[
I_x(p_1)u + I_y(p_1)v = -I_t(p_1)
\]

\[
I_x(p_2)u + I_y(p_2)v = -I_t(p_2)
\]

\[
\ldots
\]

\[
I_x(p_n)u + I_y(p_n)v = -I_t(p_n),
\]

(2.7)

where \(I_j(p_i)\) is the partial derivative of image \(I\) at point \(p_i\) with respect to \(j\) and \(p_1, ..., p_n\) are the pixels neighboring the original tracked pixel. These equations are solved with Least Mean Square estimation

\[
A^T AC = A^T B,
\]

(2.8)

where

\[
A = \begin{bmatrix}
I_x(p_1) & I_y(p_1) \\
I_x(p_2) & I_y(p_2) \\
\vdots & \vdots \\
I_x(p_n) & I_y(p_n)
\end{bmatrix},
B = \begin{bmatrix}
-I_t(p_1) \\
-I_t(p_2) \\
\vdots \\
-I_t(p_n)
\end{bmatrix},
\]

and \(C = \begin{bmatrix} u \\ v \end{bmatrix}\),

which is presented in detail by the authors in [14].

### 2.3.3 Farnebäck’s Method

A limitation to some of the other algorithms introduced to estimate motion vectors between frames is that they require the motion field to be temporally consistent, i.e. a moving object on a stationary background. If there are a lot of move-
2.3 Motion Vectors

ment in the background, theses methods have difficulty estimating displacement and produce an accurate motion field for two successive frames. Farnebäck’s method presents a solution by using polynomial expansion done spatially to approximate some neighborhood [11]. The polynomial expansion basically uses a quadratic polynomial to approximate the local signal model, \( f \),

\[
f(x) = x^T A x + b^T x + c
\]

that fits some neighborhood of pixel \( x \), where \( A \) is a symmetric matrix, \( b \) is a vector, and \( c \) is a scalar. The signal values in the neighborhood are used for a weighted least squares fit to estimate the coefficients, where the two weighting components are the same components which polynomial expansion is based on, certainty and applicability. Certainty relates to the signal values of the neighborhood, where a certainty of zero has no impact on the coefficient estimation. Applicability relates to the weight of points in the neighborhood based on position. The weighting gradually decrease with the instance from the center point.

Since each neighborhood is approximated by a quadratic polynomial, a translation between two signals \( f_1 \) and \( f_2 \) can be estimated as a global displacement, \( d \), in \( f_1 \)

\[
f_2(x) = f_1(x - d).
\]

In an example with ideal translation, the calculated coefficients in the quadratic polynomial yields

\[
A_2 = A_1, \quad b_2 = [b_1, 2A_1 d], \quad c_2 = [d^T A_1 d, b_1 d + c_1]
\]

from where the translation can be calculated from (2.12) as long as \( A_1 \) is non-singular. A more realistic example is to not interpret one signal as a single polynomial. By replacing the global polynomial in (2.10) with approximations of local polynomials ( \( A \) becomes \( A(x) \) etc.), the result will be a spatially varying displacement field \( d(x) \). This introduces new approximations for the coefficients

\[
A(x) = \frac{A_1(x) + A_2(x)}{2},
\]

\[
\Delta b(x) = \frac{1}{2} [b_2(x) - b_1(x)].
\]

With these approximations, the primary constraint can be obtained from

\[
A(x) d(x) = \Delta b(x)
\]

which is solved pointwise. To reduce the relatively high quantities of noise in the displacement field, the information of each pixel is integrated over its neighboring pixels. The integration is made by minimizing the neighborhood points with a weight function, \( \omega(\Delta x) \), formed with certainty and applicability as mentioned
in the beginning of the section. The minimum value $e(x)$ is given by

$$e(x) = \left[ \sum \omega \Delta b^T \Delta b \right] d^T(x) \sum \omega A^T \Delta b]. \quad (2.17)$$

### 2.4 Tukey Windowing

In one of the proposed methods for measuring respiratory frequency mentioned in Section 1.5, calculations will be done in the frequency domain. Spectral leakage occurs when using Fourier transform on raw data, i.e. spectral information ends up in the wrong frequency. Window functions are often used in signal processing to reduce spectral leakage. It is zero-valued outside its interval, and shaped like a chosen mathematical function inside, often symmetric around the middle of the interval. The Tukey window is a type of window function based on a cosine lobe of width $\alpha N/2$ convolved with a rectangular window of width $N(1 - \alpha N/2)$. $N$ denotes the length of the resulting window and $0 < \alpha < 1$ is the factor deciding how steep the filter is. $\alpha = 0$ results in a rectangular window and $\alpha = 1$ results in a full cosine lobe (Hann window). The Tukey window is mathematically described for all $0 < \alpha < 1$ as $\omega[n] = 1/2 - \cos(\pi n / \alpha N)$ for $0 \leq n < \alpha N/2$, $\omega[n] = 1$ for $\alpha N/2 \leq n \leq N/2$, and $\omega[N - n] = \omega[n]$ for $0 \leq n \leq N/2$. Figure 2.4 shows four Tukey windows with varying $\alpha$.

![Figure 2.4: Five examples of Tukey windows with window length $N = 128$, for $\alpha = 0, \alpha = 0.25, \alpha = 0.5, \alpha = 0.75, \alpha = 1$.](image-url)
2.5 Point-Mass Filter

The point mass filter is a way to approximate a state using recursive estimation, often used for navigation. Although the approach is similar to the Kalman filter, the point-mass filter has no restrictions of linearity in the state, and more importantly, no restrictions on the noises acting on the model to be Gaussian. The underlying state space model is given by

\[ x_{t+1} = x_t + u_t + v_t \]
\[ y_t = h(x_t) + e_t \]  \hspace{1cm} (2.18)

where \( v_t \) and \( e_t \) are mutually independent white processes, \( u_t \) denotes the estimated movement between two measurements and \( h(\cdot) : \mathbb{R}^2 \rightarrow \mathbb{R} \) is the function for the terrain elevation map (when used in navigation) and \( x_t \) is the state to be estimated. The approximation of the state is computed by iterating the Bayesian solution [15] given by equation (2.19)-(2.23),

\[ \alpha = \int_{\mathbb{R}^2} p_{e_t}(y_t - h(x_t))p(x_t|Y_{t-1})dx_t \]  \hspace{1cm} (2.19)
\[ p(x_t|Y_t) = \alpha^{-1} p_{e_t}(y_t - h(x_t))p(x_t|Y_{t-1}) \]  \hspace{1cm} (2.20)
\[ \hat{x}^{MS}_t = \int_{\mathbb{R}^2} x_t p(x_t|Y_t)dx_t \]  \hspace{1cm} (2.21)
\[ C_t = \int_{\mathbb{R}^2} (x_t - \hat{x}^{MS}_t)(x_t - \hat{x}^{MS}_t)^T p(x_t|Y_t)dx_t \]  \hspace{1cm} (2.22)
\[ p(x_{t+1}|Y_t) = \int_{\mathbb{R}^2} p_{v_t}(x_{t+1} - u_t - xt)p(x_t|Y_{t-1})dx_t. \]  \hspace{1cm} (2.23)

where \( p(\cdot) \) denotes the distribution, \( C_t \) denotes the covariance, and \( \hat{x}^{MS}_t \) denotes the estimation done with minimum mean square error. Without diving too deep into the Bayesian solution, the foundation of the point-mass filter is built on the iterating process with one approximation in that \( x_t(k), \ k = 1, 2, ..., N \) where \( N \) grid points in \( \mathbb{R}^2 \) are chosen. This means that the integrals from equation (2.19)-(2.23) can be approximated by a finite sum know as Riemann sums:

\[ \int_{\mathbb{R}^2} f(x_t)dx_t \approx \sum_{k=1}^{N} f(x_t(k))\delta^2 \]  \hspace{1cm} (2.24)

where \( \delta \) is the distance between the grid points \( N \). Rewriting the Bayesian solution we now get
\( \alpha = \sum_{k=1}^{N} p_{e_t}(y_t - h(x_t(n)))p(x_t(n)|Y_{t-1})\delta^2 \) \hspace{1cm} (2.25)

\[
p(x_t(k)|Y_t) = \alpha^{-1} p_{e_t}(y_t - h(x_t(k)))p(x_t(k)|Y_{t-1})
\] \hspace{1cm} (2.26)

\[ \hat{x}_t^{MS} = \sum_{k=1}^{N} x_t(n)p(x_t(k)|Y_t)\delta^2 \] \hspace{1cm} (2.27)

\[ C_t = -\sum_{k=1}^{N} (x_t - \hat{x}_t^{MS})(x_t(n) - \hat{x}_t^{MS})^T p(x_t(n)|Y_t)\delta^2 \] \hspace{1cm} (2.28)

\[ x_{t-1}(k) = x_t(k) + u_t, \quad k = 1, 2, ..., N \] \hspace{1cm} (2.29)

\[ p(x_{t+1}|Y_t) = \sum_{k=1}^{N} p_{v_t}(x_{t+1}(k) - u_t - xt(n))p(x_t(n)|Y_{t-1})\delta^2. \] \hspace{1cm} (2.30)

Where equation (2.29) is a consequence of the discretization. The approach of the algorithm is as N. Bergman describes it [15]:

- Discretization of the state space. Limit the range of values of \( x_t \) from the continuum \( \mathbb{R}^2 \) to a finite number of levels.

- Numerical approximation of the integrals. Replace the integrals with Riemann sums over finite intervals.

- Probability region equivalent. Derive the state space into regions and express the probability of being in each region. Use this probability as a weight in each region.

- Piecewise constant approximation of the posterior. Numerically approximate the posterior as a sum of weighted and shifted indicator functions.

- Nyquist approach. Assume that the posterior is band-limited, i.e. has an upper bound on the frequency of spatial variation. Sample the function spatially and update these samples.
This chapter covers the methodology used to answer the questions proposed in Section 1.3. It includes how the data collection and evaluation were done, the filtering method used on the data and the image and signal processing methods used to obtain the optical flow and the signal used to estimate breathing frequencies from the data. The two methods developed are presented as "Breathing Movement Zero-Crossing Counting" and "N-Point Fast Fourier Transform".

3.1 Data Collection & Validation

To develop and test the algorithms, good data sets are needed. The recorded data were collected by the authors, and to ensure a versatile data set, some different scenarios and locations were used. This section covers the two data collections and the data validation made during the project.

3.1.1 Initial Data Collection

The initial data collection was done at Vreta Naturbruksgymnasium, where dairy cows and calves were filmed with three different cameras. To achieve good video quality, in a sense of image processing, the cameras had to be stationary and the different takes were made so that variations in lighting conditions was kept as low as possible. The three different cameras were mounted on a tripod and placed as close to each other as possible to minimize differences in the videos. For every location and target, a couple of different resolutions and frame rates were captured with the Intel RealSense and the ESP camera.

The fully grown cows were recorded while eating at location (a) in Figure 3.1. This location was used because the cows were standing still for a couple of min-
utes at a time while eating. Movements in the abdominal region, due to the chewing, lead to difficulties in manually observing the respiratory rate. Since the future algorithm for calculating respiratory rate needs to be compared with the true respiratory rate, this location was deemed unusable for data collection.

A new location was then selected, where the younger and smaller cows were the new recorded subjects, see location (b) in Figure 3.1. This location had some altitude difference between the cameras and the cows, where the cameras were at a higher position and therefore gave a better overview. Additionally, the location ensured there would not be any obstacle between the cow and the camera. These younger cows were a little bit more careful in approaching the cameras, which led to less movement in the videos.

The final location of the first recording session was the milking robot, where the cows are forced to stay relatively still during the process, see location (c) in Figure 3.1. Initially, this seemed like a good location to record the cows, but there were some problems here as well. One of the problems was that the cows are fed during the milking process which lead to the same problem as at the first location where the eating creates breathing-like movements. Another problem was that a milking robot is a big piece of machinery, and some of it blocked the line of sight to different parts of the cow, depending on the size of the cow. To avoid this, the Basler camera was mounted on a pillar to get free sight to the cow. The two other cameras did not have long enough power cords to reach this position. Thus, this data set mostly contains recordings with the Basler mounted on the pillar.

### 3.1.2 Data Validation

To simplify the algorithm development, the collected data were examined. The goal was to remove the data where the line of sight to the cow was obscured or some other disturbance appeared in the shot. This was done manually by observing the recorded videos.

The data from the ESP-CAM had some major disturbance in it, caused by its own LED diode. A case for the ESP-CAM had been 3D-printed to make it possible to mount the camera on the same tripod as the other cameras. The opening on the case for the diode was covered with tape, which led to light emitting from the opening of the camera which disturbed the data enough and made the recordings unusable.

The RealSense camera recorded on three channels: one depth channel, one IR channel and one RGB channel. This required too much processing power or bandwidth and resulted in low quality and frame rate on every channel. The frame rate is important to maintain at the desired level to detect small movement changes in the videos. Since this was not the case, the IR channel was not used in the second recording session. The processing power required for the IR channel was insted used to achieve higher quality and higher frame rate for the RGB and depth channel. The RealSense camera was not able to get any recordings from above the milking robot due to its shorter cord.
The Basler camera videos taken above the milking robot were overexposed, leading to difficulties in observing the cow. However, the frame rate and resolution were stable. Thus, the Basler camera was considered to be the best option of the three cameras.

### 3.1.3 Secondary Data Collection

The secondary and final recording session was made to obtain more valid data for the algorithm development. The main recording position was the pillar by the milking robot, location (d) in Figure 3.1. A quick setup was built to reach out from the pillar and get a view directly over the cows. This, together with changed exposure settings for the camera resulted in data that was not overexposed. While the Basler camera filmed the milking robot, the Intel RealSense was used to film fully grown cows that were laying down and resting, locations (e) and (f) in Figure 3.1.

![Figure 3.1: Field of view from the six different camera location. Locations (a)-(d) in gray scale are from the Basler camera and location (e)-(f) in color are from the RealSense’s RGB camera.](image)
3.2 Breathing Movement Zero-Crossing Counting

This section presents the Breathing Movement Zero-Crossing Counting algorithm methodology, one of two suggested strategies for estimating the respiratory rate. The other strategy is the N-Point Fast Fourier Transform algorithm, presented in Section 3.3. The aim is to count the zero-crossings on a signal that corresponds to the respiratory movements of the cow. These movements are extracted motion vectors from the optical flow, calculated with the Lucas-Kanade method, and presented in Section 3.2.2. However, the optical flow calculation is relatively computational heavy. Therefore, to reduce the region of the image that the optical flow algorithm is applied to, a suggested object tracker is presented in Section 3.2.1.

Another problem is that the calculated movements from the optical flow are not entirely related to the respiratory cycle. Therefore, a local median movement is calculated and subtracted to yield a signal corresponding to the respiration, presented in Section 3.2.3. On the other hand, this limits the algorithm to a specific camera location where the camera needs to be placed above the targeted cow, like location (d) in Figure 3.1. Next, the obtained signal is bandpass filtered with filter parameters decided with a Fourier transform approach, presented in Section 3.2.4. Finally, the zero-crossings algorithm to estimate the respiratory rate is presented in Section 3.2.6.

3.2.1 ROI Tracking

A Region Of Interest (ROI) tracking algorithm was implemented to reduce the calculations when calculating the optical flow of a moving object without a trained network of object detection for cows. This algorithm had the requirement of operating in real-time, thus an online multiple instance learning (MIL) algorithm was used [16]. A MIL tracker trains and updates its classifier continuously to detect the object in the current frame. Therefore, it is considered to be an online tracker.

The workflow of the tracker can be seen in Figure 3.2. In the first frame, a region of interest surrounding the object is manually initialized and image features are calculated, creating a features pool. Next, the classifier is initialized with the feature that has the highest probability of being found in the patch, which in the first frame is the region of interest. The probability can be written as 

\[ p(y = 1 | x), \]

where \( y \) is a binary variable that represents the object’s presence in the patch \( x \).

For the example from Figure 3.2, the classifier is initialized with the features of the smiley face’s mouth and eyes. In the second frame in Figure 3.2, the smiley face’s mouth is no longer visible. A set of image patches, \( X^s \), that neighbors the current tracker location with radius, \( s \), are cropped out

\[ X^s = x | s > ||l(x) - l^s_{t-1}||, \]

where \( l(x) \) and \( l^s_{t-1} \) denotes the location of patch \( x \) and the tracker at time \( t-1 \) respectively. Since the mouth is no longer visible in frame 2, the current classifier is updated with a feature from the extracted bag. The eyes are visible in every patch.
in the bag. Therefore, the feature corresponding to the eyes is a valid classifier. Finally, the location of the tracker is updated with a greedy strategy

$$l_t^* = l(\arg\max_{x \in X} p(y = 1|x)),$$

where $t$ is the current timestamp. This returns the targeted object’s location at every timestamp and reduces the required calculation of the optical flow, presented in the following section.

<table>
<thead>
<tr>
<th>Frame 1</th>
<th>Init Classifier</th>
<th>Features pool:</th>
<th>Frame 2</th>
<th>Update classifier</th>
<th>Features pool:</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Frame 1" /></td>
<td><img src="image2" alt="Init Classifier" /></td>
<td><img src="image3" alt="Features pool" /></td>
<td><img src="image4" alt="Frame 2" /></td>
<td><img src="image5" alt="Update classifier" /></td>
<td><img src="image6" alt="Features pool" /></td>
<td><img src="image7" alt="Result" /></td>
</tr>
</tbody>
</table>

ROI init: Classifier = \{\(\cdot\)\}

Extract bag: Classifier = \{\(\cdot\)\}

**Figure 3.2: MIL tracking workflow, inspired from [16]**

### 3.2.2 Optical Flow with Lucas-Kanade algorithm

As presented in Section 2.3.2, the Lucas-Kanade method typically uses pixels in the images that are easy to track, like corners where the foreground and background are easily separated. However, since the hair color of a cow can be monochrome without any patterns, one can not expect good results when trying to track pixels using this strategy. So instead, a universal method was implemented, where every pixel to track was pre-decided and updated at every frame together with the ROI tracking algorithm from Section 3.2.1. The tracked pixels form a grid over the object, and the number of tracked pixels directly relates to the density of the optical flow. Simulations with varying number of pixels to track were performed, and the results are presented in Section 4.2.1.

Using a tracker reduces the computations compared to an algorithm without tracking since the optical flow is only calculated for the object and not the surrounding. The tracked pixels and ROI tracker are visualized in Figure 3.3.
As mentioned in Section 2.3.2, the Lucas-Kanade algorithm calculates the optical flow for every pixel neighboring the original pixel to track. The size of the box that represents the area containing the neighboring pixels is essential for the computation rate of the algorithm and the noise impact. The larger the box, the slower computations but less noise impact. Deciding the specifications of the box was done manually by simulating with different sizes and comparing the results with computation time. These simulations are presented in Section 4.2.1.

To decide which part of the cow is moving, the tracked pixels are divided into nine cells overlaying the region of interest, see Figure 3.4. This leads to that every tracked pixel has a cell index that corresponds to a specific part of the cow, and since the pixels to track are re-created at every frame with the ROI tracker, these cells will always correspond to the same part of the observed cow.

The cells from Figure 3.4 consist of motion vectors that represent the velocity of every tracked pixel. These motion vectors have two components, as described in Section 2.3, where $u$ and $v$ represent horizontal and vertical velocity, respectively. However, the algorithm is limited to operate at location (d) from Figure 3.1. Thus, the cow is always rotated in the same direction. This leads to the majority of the
respiratory movements being horizontal, and the cow’s vertical movements are therefore not of interest. Further, the number of motion vectors in every cell can differ if the tracked pixel is not found in the successive frame, leading to some uncertainties in the true velocity of the cell. This uncertainty was minimized by calculating a median horizontal velocity in every cell

\[ u_{T,i,j} = \text{med}(\{u_{T,i,j,1}, \ldots, u_{T,i,j,N-1}, u_{T,i,j,N}\}) \]  

(3.3)

where \( T, i, j \) and \( N \) represents the current timestamp, row index, column index and number of motion vectors respectively. Further, the median velocities are calculated for every timestamp, creating a signal

\[ u(t)_{i,j} = [u_{0,i,j}, \ldots, u_{T-1,i,j}, u_{T,i,j}] \]  

(3.4)

with median velocity over time for all nine cells. These signals represents the cow’s movement and are processed in the upcoming section.

### 3.2.3 Extracting Breathing Movements

In the ideal case, where the cow is standing still and not eating, all calculated signals from Section 3.2.2 should correspond to a breathing cycle. Over an arbitrary time interval, the signals would resemble sinus-shaped signals. Since all movements from the cow are calculated, a signal that seems to be sinusoidal does not need to represent a breathing cycle. To eliminate movements not part of the breathing, the cow’s horizontal local median velocity was subtracted from the signal \( u(t)_{i,j} \) at every timestamp. This limits the algorithm to only function when both sides of the cow are visible, as they always are at location (d) in Figure 3.1. However, this creates the signal

\[ y(t)_{i,j} = u(t)_{i,j} - u(t)_i, \]  

(3.5)

that represents the respiratory movements of the cow in cell \((i,j)\). The median velocity, \( u(t)_i \), for row \( i \) is calculated as in equation (3.4)

\[ u(t)_i = [u_{0,i}, \ldots, u_{T-1,i}, u_{T,i}] \]  

(3.6)

where \( u_{T,i} \) represents the median velocity in row \( i \) at time \( T \). This is visualized in Figure 3.5, where the blue, red and magenta curves represent the median velocities \( u(t)_{i,j}, u(t)_i \) and \( y(t)_{i,j} \), respectively.
As mentioned in Section 2.1, the respiratory rate of a healthy cow is between 15-35 BPM. However, a stressed or sick cow could reach as high as 89 BPM. The frequencies of interest are therefore an interval between 15 and 89 BPM. An initial bandpass filter, \( h_{BP1} \), with cutoff frequencies at 0.1 Hz and 1.63 Hz was applied to the signal \( y(t)_{i,j} \),

\[
y(t)_{i,j}^{BP} = y(t)_{i,j} \ast h_{BP1}.
\]  
(3.7)

The filtered signal \( y(t)_{i,j}^{BP} \) is visualized as the the black line in Figure 3.6.

**Figure 3.6:** The black line \( y(t)_{i,j}^{BP} \) corresponds to the BP-filtered signal, and is the estimated breathing movement of the cow.

For the example in Figure 3.6, the bandpass filtered black line indicates a relatively symmetric signal, with one very distinctive fundamental frequency. How-
ever, for some subjects, the extracted breathing movement could be represented with a signal containing harmonics to the fundamental frequency. Thus there is a need for a second bandpass filter with stricter cut-off frequencies. The calculations of these frequencies are presented in the next section.

### 3.2.4 Filter Parameter Estimation

The calculated breathing movements $y(t)_{i,j}^{BP}$, could in some cases be observed as a bit jerky and asymmetric. This, together with the movements in the abdominal region of the cow that is not part of the breathing, results in a signal $y(t)_{i,j}^{BP}$ shown in Figure 3.6, which contains harmonics to the fundamental frequency. This is visualized in Figure 3.7, where multiple frequency components can be observed. However, if the operations in Section 3.2.3 are successful, the fundamental frequency would correspond to the frequency of the respiratory cycle. Since the cut-off frequencies of the initial bandpass filter $h_{1}^{BP}$ is not strict enough to discard the overtones, a respiratory rate can't be calculated with high certainty from the signal $y(t)_{i,j}^{BP}$.

![Figure 3.7: A visual representation of the extracted and bandpass filtered signal $y(t)_{i,j}^{BP}$ containing multiple frequency components.](image)

To identify the fundamental frequency, a second bandpass filter, $h_{2}^{BP}$, is used. The cut-off frequencies for this filter are automatically set by frequently observing the power spectrum $P(w)$ of the Fourier transformed signal $Y(w)_{i,j}^{BP} = \mathcal{F}\{y(t)_{i,j}^{BP, ZP}\}$

$$P(w) = |Y(w)_{i,j}^{BP}|^2,$$

where the signal $y(t)_{i,j}^{BP, ZP}$ is the signal $y(t)_{i,j}^{BP}$ zero-padded with its length to increase frequency resolution, as described in appendix A.1.

The power spectrum in Figure 3.8 shows the frequency components found in the signal in one specific cell. To decide which frequency component corresponds
to the fundamental frequency, an estimate is calculated for the highest and second highest peak at frequencies $f_1$ and $f_2$, respectively. These frequencies have to be separated with a minimum difference of 0.1 Hz to be treated as different frequency components. Hence, the peaks contain a small interval of frequencies, not just one. The two estimates,

$$E_{f_1} = \frac{P(f_1)}{P(w)} \quad \text{and} \quad E_{f_2} = \frac{P(f_2)}{P(w)},$$

are the quotas between the power of the peak and the power of the signal.

**Figure 3.8:** Power spectrum of the Fourier transformed signal corresponding to signal $\check{y}(t)_{i,j}^{BP}$ seen in Figure 3.7.

The fundamental frequency, $f_f$, in the signal $\check{y}(t)_{i,j}^{BP}$ from Figure 3.7 is estimated to be the frequency at $f_1$ in Figure 3.8. To specify the certainty of the estimation, a quality measurement $Q$ is calculated. This is a measure of how certain the algorithm is for estimating the filter parameters of $h_2^{BP}$, and is done for every cell from Figure 3.4. The quality is calculated as

$$Q = 1 - \frac{E_{f_2}}{E_{f_1}},$$

where the quality approaches one in the ideal case with $P(f_1) >> P(f_2)$ and zero in the case where $P(f_1) = P(f_2)$.

The data in the cell that corresponds to the highest value of $Q$ will be the estimation data in the following section. The remaining cells will be disregarded until the next measurement is done.
3.2.5 Signal Estimation

The obtained fundamental frequency, from Section 3.2.4, sets the cut-off frequency for the bandpass filter, \( h_{BP}^2 \), with a low-cut frequency at \( f_f - 0.1 \) Hz and a high-cut frequency at \( f_f + 0.1 \) Hz. On the other hand, the obtained signal is estimated with collected data over an arbitrary time interval, as mentioned in Section 3.2.3. Therefore, an opportunity is provided to use forward and backward offline filtering, which results in a zero-phase signal. The filter is applied in three steps. Firstly, \( y(t)_{i,j}^{BP} \) is passed in the forward operation

\[
v(t) = h_{BP}^2 \ast y(t)_{i,j}^{BP}.
\]  

(3.11)

Then, for the backward operation, \( v(t) \) is flipped to obtain \( v(-t) \) and passed through the filter

\[
w(t) = h_{BP}^2 \ast v(-t).
\]  

(3.12)

Finally, the signal \( \hat{y}(t)_{i,j} \) is obtained by flipping \( w(t) \)

\[
\hat{y}(t)_{i,j} = w(-t),
\]  

(3.13)

and is visualized as the green line in Figure 3.9.

![Figure 3.9: The green line is the extracted and applied bandpass filtered signal \( h_{BP}^2 \), which corresponds to the breathing movement of a cell as in equation (3.7)](image)

3.2.6 Zero-crossing Algorithm

To estimate the frequency of the signal \( \hat{y}(t)_{i,j} \), a zero-crossing algorithm is used. Since the two most distinctive frequencies from Figure 3.8 have a limitation on the minimum difference in frequency, the second-highest peak could be located near the highest peak and therefore rejected. However, when filtering with the bandpass filter \( h_{BP}^2 \), this rejected peak will still be included. Thus, the signal \( \hat{y}_{i,j} \) can contain more than one distinctive frequency component.

The median difference in time between two adjacent zero-crossings is calculated
to minimize the estimation error and the effect of imperfect measurements. The
difference in time between two zero-crossings equals half a breathing cycle time
since either an intake or an exhaust is performed during the time interval. The
calculated zero-crossings are shown in blue in Figure 3.10 and the estimated res-
piratory frequency, in breaths per minute, is calculated as

\[ BPM = \left( 2 \times \text{med}(zc_2 - zc_1, zc_3 - zc_2, ..., zc_n - zc_{n-1}) \right)^{-1} \times 60, \]

where \( zc_n \) and \( zc_1 \) are the timestamps at the last and first zero-crossing respectively.

3.3 Frequency Estimation with N-Point Fast Fourier Transform

This section presents the algorithm methodology for the N-point Fast Fourier
Transform method, one of two suggested strategies for estimating the respiratory
rate. Unlike the Breathing Movement Zero-Crossing Counting algorithm, this
method proposes a solution without the requirement to know the spatial loca-
tion of the subject. The signals are mostly processed in the frequency domain,
unlike the first method that does the processing in the time domain. The algo-

\[ y(t)_{i,j}, \text{Zero-crossings} \]

\[ 0 \quad 2 \quad 4 \quad 6 \quad 8 \quad 10 \quad 12 \quad 14 \quad 16 \quad 18 \quad 20 \]

\[ \text{Velocity [pixels/frame]} \]

\[ -0.3 \quad -0.2 \quad -0.1 \quad 0 \quad 0.1 \quad 0.2 \quad 0.3 \]

\[ 0 \quad 2 \quad 4 \quad 6 \quad 8 \quad 10 \quad 12 \quad 14 \quad 16 \quad 18 \quad 20 \]

\[ \text{Time [s]} \]

\[ \text{Velocity [pixels/frame]} \]

\[ zc_9 \quad zc_{10} \]

Figure 3.10: Calculated zero-crossings from bandpass filtered signal \( y(t)_{i,j} \)
from Figure 3.9

This will all be presented together with pre-processing meth-
ods from Section 2.2 and a filter inspired by the point-mass filter proposed in
Section 2.5, designed to autonomously determine which estimated frequency is
the breathing frequency.
3.3 Frequency Estimation with N-Point Fast Fourier Transform

3.3.1 Pre-processing

The raw data used to implement and test the two methods of estimating the breathing frequency in this section is merely a sequence of digital images collected with a fixed frame rate. A well analyzed pre-processing procedure is important to prepare the raw data set for future processes and to optimize the computation time of the system as a whole. An easy way to save time is to reduce the amount of information that is processed. In this case, it is done by downsampling the images and sub-sampling the image feed. There are two straightforward methods. Downscaling simply resizes an image of size \((m \times n)\) by a factor \(k\) by creating a new pixel value \(P_{DS}\) from the mean of the corresponding \((k \times k)\)-area of the original image. The downscaled image size is thereby reduced to \((\frac{m}{k} \times \frac{n}{k})\).

3.3.2 N-Point Fast Fourier Transform

One of the methods used to estimate breathing frequencies from the collected data is a solution based on an N-point Fast Fourier Transform process. The process uses the optical flow of an N long sequence of frames, where each pixel’s motion vector in every frame creates a complex value. The estimated optical flow from either Farnebäck’s algorithm or Horn Schunk’s algorithm, presented in Section 2.3.3 and Section 2.3.1, return motion vectors, \((u, v)\), as a pixels velocity along the x-axis respectively the y-axis as a frame. With \(N\) successive number of these frames a 3D-matrix of complex motion vector values is created by using each pixel’s x-velocity as the real number and y velocity as the imaginary number, creating the complex time-domain velocity \(Z[i, j, n]\) described in equation (3.15).

\[
Z[i, j, n] = U[i, j, n] + iV[i, j, n] \tag{3.15}
\]

The pixel velocity at row position \(i\) and column position \(j\) in the x-axis is denoted as \(U[i, j, n]\) and the velocity in the y-axis as \(V[i, j, n]\), and \(n\) denotes the position along the z-axis illustrated by Figure 3.11. To minimize computational complexity, the frames of velocities are downscaled by a factor of 16 in the same way as described in Section 3.3.1, creating "boxes" with mean velocities in \(Z[i, j, n]\).
Zero-padding with $N$ number of zeros was used to improve the frequency resolution, see Appendix A.1, doubling the length of the z-axis to $2N$. To further improve the quality of the FFT-process, the 3D-matrix is masked with a Tukey-window, $w_{tu}[n]$, along the z-axis, described in Section 2.4. The next step was the calculations of the FFT, presented in Appendix A.2. Fast Fourier transform is performed on each frame of the time-domain velocity, i.e. for all $n$ of $Z[i, j, n]$. The result is a sequence of values depending on temporal frequency instead of discrete time. The Fourier transform gives an output twice as long as the input and symmetric around the frequency at $n = 0$. The output thus has the length $4N$. Therefore no useful information is lost when only using $n > 0$, and since breathing frequencies are low, only the first half of $n < 0$ output is of interest. To summarize, all breathing frequencies are found in one forth of the output, a region illustrated by "***" in Figure 3.12.

**Figure 3.11**: Illustration of the 3D-matrix $Z[i, j, n]$ with complex motion vector values. The notations $u + iv$ represent $U[i, j, n] + iV[i, j, n]$ at the specific position $[i, j, n]$. 
Figure 3.12: Example of a random output signal with length $4N$ in red, and the four symmetric regions. The region of interest is defined as "***".

The values after the transform are still complex. To get the spectral information needed, the absolute value of the region is calculated. These steps are given by

$$Z_{zp,tu} = Z_{zp} w_{tu}$$

(3.16)

$$\hat{Z} = \mathcal{F}\{Z_{zp,tu}\}$$

(3.17)

$$\hat{Z}_{\text{abs}} = |\hat{Z}[0 : \frac{4N}{4} - 1]|$$

(3.18)

where $Z_{zp}$ is the zero-padded version of $Z$. All "boxes" for a specific $[i,j]$ will then be a $N$ long "3D-box" contain spectral information from that position, $[i,j]$, from the last $N$ frames. From the spectral information, the maximum of each "3D-box" represents a higher intensity of a specific frequency, see Figure 3.13 and 3.14.

Figure 3.13: Figure of spectral information from an index of $\hat{Z}_{\text{abs}}$
The index of that maximum value along the frequency-axis is calculated with
\[
M_{\text{ind}}[i, j] = \arg \max_{[n]} \hat{Z}_{\text{abs}}[i, j, n].
\] (3.19)
as is illustrated by Figure 3.14. For most indices, the location of the maximum
values, \(M_{\text{ind}}[i, j]\), are found in \(n = 0\) which gives a resulting matrix \(M_{\text{ind}}[i, j]\)
filled with mostly zeros.

If an index other than \(n = 0\) is given, a non-zero frequency has been found. What
frequency has been found is then calculated by multiplying with the sampling
frequency and dividing with the length \(N/2\). Thus, the resulting frame of found
frequencies, \(F_{\text{Hz}}[i, j]\), is given by
\[
F_{\text{Hz}}[i, j] = \frac{M_{\text{ind}}[i, j] \cdot f_s}{N/2}.
\] (3.20)
Where \(F_{\text{Hz}}[i, j]\) is given in Hertz.

The algorithm can be explained by the following steps:

1. Calculate \(N\) frames of optical flow and save as a complex 3D-matrix as in
equation (3.15).
2. Downscale to \((16 \times 16)\) "boxes".
3. Use Zero-padding and mask the result with a Tukey window along the z-
axis like (3.16).
4. Apply the Fast Fourier Transform to each "box" along z-axis and save the
first fourth of the result as equation (3.17) and (3.18).
5. Identify the index of the maximum value for each "box" as in equation
(3.19).
6. Multiply $M_{\text{ind}}[i, j]$ with Nyquist frequency and $N/2$ to get a frame of frequencies, $F_{\text{Hz}}[i, j]$.

7. Discard frame at position $n$ from $Z(i, j, n)$ and shift all remaining frames one step, calculate optical flow and item 2 on new image and save in position 0. Start over at item 3.

### 3.3.3 Neighborhood Weighing

The matrix of found frequencies, $F_{\text{Hz}}[i, j]$, does not only includes the breathing frequency of the subject. Other movements and noise from the motion vector estimation can also give frequency results. A weight system based on the neighboring frequencies is used to help decide which frequency corresponds to the breathing. It is made with one simple assumption; If a found frequency represents the breathing rate, nearby "boxes" should also be of that same frequency. Thus, a radius-based weight system where every found frequency gets a weight equal to the number of boxes with matching frequencies inside the radius, including the box where the weight is calculated. This is illustrated in Figure 3.15

![Figure 3.15: Illustration of a matrix of frequencies and a decided weight](image)

where every non-zero "box" has a frequency a corresponding weight. When all weights are calculated, the weighted frequencies create a histogram where all weights for that specific frequency are added so that the list of $k$ frequencies $f_k = [0, 1, 2, ..., k]$ have a matching histogram of added weights $w[k] = [0, w(f_1), w(f_2), ..., w(f_k)]$. Frequencies in $f_k$ not included in $F_{\text{Hz}}[i, j]$ are weighted as 1, which gives $w[k]$ a weight for all integers in $f_k$.

### 3.3.4 Point-Mass Inspired Filter

For further improvements of the estimation of the frequency representing the breathing rate, an approximation inspired by the point-mass filter shown in Sec-
tion 2.5 is used. The weighted histogram from previous Section 3.3.3, \( w[k] \), is used to compute the posterior, \( \alpha \). It is scalar multiplied with the prior, \( P_{n|n-1} \), which in the first cycle is a normalized equal distribution over the frequency range

\[
\alpha = w(k)P_{n|n-1}.
\] (3.21)

Further, to create a distribution from the histogram, the result is convolved with a kernel of one-dimensional Gaussian distribution, \( x_G[l] \), from equation (2.1). The kernel is simply a vector with its center in 0 and of width \( l_N/2 \), i.e. \( x_G[l_N] = [-l_N/2, ..., -1, 0, 1, ..., l_N/2] \). The length of the kernel, \( l_N \), decides the range of the smoothing effect from the convolving

\[
\alpha_G = \alpha \ast G(x_G[l_N]),
\] (3.22)

and since the convolving has an output longer than the input posterior, a shift is performed to regain the original size. After the shift, the distribution is normalized to be used as the prior of the next cycle

\[
P_{n|n-1} = \frac{\alpha_G}{\|\alpha_G\|}.
\] (3.23)

Next, the mass point, \( \hat{x}_t^{MS} \), of the distribution is used as the estimation of the breathing frequency and is calculated by

\[
\hat{x}_t^{MS} = \sum_{s=1}^{k} f_k P_{n|n-1},
\] (3.24)

where \( f_k \) is the same list of frequencies as in 3.3.3. Finally, as in 2.5, the covariance error, \( C_t \), can be obtained from the mass point

\[
C_t = \sum_{s=1}^{k} (f_k - \hat{x}_t^{MS})(f_k(n) - \hat{x}_t^{MS})^T P_{n|n-1}
\] (3.25)

and used to quantify the precision of the estimation. This cycle can be described similar to the steps in N. Bergman’s work referenced in Section 2.5, but with a slightly simplified 1-d perspective. The steps for a cycle are:

1. Build a Gaussian convolution kernel, \( x_G \), with length \( l_N \) as in equation (2.1).
2. Categorize the frequencies from \( F_{Hz}[i, j] \) to the nearest integer and apply the neighborhood weighing presented in Section 3.3.3.
3. Compute the posterior by spatial multiplication of prior and weighted frequency histogram as in equation (3.21).
4. Perform the convolution between the posterior and the Gaussian kernel as in equation (3.23).
5. Normalize the result and save as next cycles prior.
6. Estimate the breathing frequency by computing \( \hat{x}_t^{MS} \) as in equation (3.24).
7. Compute the error covariance for the estimation as in equation (3.25).

8. Wait for next $w(k)$ and continue at item 2 above.

The filter uses earlier frequency distributions to withhold sudden frequency spikes relatively far from estimated breathing frequency. Hence, no matter the weight, a new frequency needs to be consistent over several samples to be considered a breathing frequency. The length of the Gaussian kernel, $I_N$, decides the system’s response time when the estimated breathing rate changes slightly, although with the length increases, the risk of sudden frequency spikes or other noise magnifies. Figure 3.16 illustrates step 3-5 where a hypothetical sudden frequency spike emerges in $w(k)$.

**Figure 3.16:** Step 3-5 in the point-mass inspired filter when a spike of noise emerges in $w_N$

The error covariance is calculated to give a certainty on the estimation, which gives the opportunity to discard the estimation as a breathing frequency while it still is used in the calculations of the next estimation.
This chapter presents the results from the respiratory rate estimation techniques presented in Chapter 3. The results from the N-point FFT-method are presented in Section 4.1 with \( N = 256 \) and the Breathing Movement Zero-Crossing Counting-method is presented in Section 4.2 as well as the intermediate results from each method’s sub-systems.

### 4.1 N-Point Fast Fourier Transform

The following section presents the results from the N-point FFT-method described in Section 3.3.2 in terms of the estimated respiratory rate over the recorded data and compares it to the respiratory rate ground truth. An analysis will also be made of how often the estimated frequency is a reliable respiratory rate. All simulations are done with \( N = 256 \).

#### 4.1.1 Motion vectors result

As mentioned in Section 3.3.2, two methods of estimating motion vectors were tested when implementing the N-point fast Fourier transform-solution. The two methods were Farnebäck’s algorithm (see Section 2.3.3) and the Horn-Schunk algorithm (see Section 2.3.1). One of the most vital calculations of the process is the motion vector estimation to create the optical flow. The estimation exploits much of the system’s computational power, and the quality of the output of the estimation is vital for the accuracy of the system in total. When reviewing the results of the motion vectors, there are some key points to have in mind. A first evaluation consists of to what degree the method works on the collected data, using visual evaluation. Initially, it’s hard to know how a method works in a specific envi-
ronment, especially the variations in lighting of the different recording locations from Section 3.1. Key point 1 is thereby to evaluate the amount of visual noise from the estimations. Further on, the amount of motion vector data needed for the system to work is another key aspect. Therefore, key point 2 is the resulting motion vectors from varying image resolution is of interest. For the system to work on limited hardware in real-time while still having the accuracy in distinguishing relatively slow movements like breathing, a moderate resolution setting is sought. Finally, both the Farnebäck and Horn & Schunk algorithms have parameter settings where the result of different configurations are presented as key point 3 in the coming section.

With these key points as a foundation for the motion vector performance, three correlated types of results are presented in this section. The input is two consecutive images from the data set. One of the images is displayed in Figure 4.1. This image pair will be used for the results of different resolutions.

![Figure 4.1: One of the images used as input for the experimentation of resolution for motion vectors](image)

The configurations from Farnebäck’s and Horn-Schunk’s methods and a few results are presented in Section 4.1.1.

**Farnebäck’s method**

The Farnebäck method was performed with the three key points in mind, first with input from the data set with resolution (256 × 136), (512 × 272), (1024 × 544), and (2048 × 1088) where remaining configurations are set as default. Parameters set for a run are the number of iterations and the polynomial order, where the default settings are **Iterations** = 15 and **PolyN** = 5. The Farnebäck method uses every pixel in the image to estimate motion vectors; therefore, the number of
pixels corresponds to the amount of calculations. The results for resolution are shown in Figure 4.2, where the images of the motion vector fields show a brighter nuance where the pixel movement is higher and black where the pixel is still. The configurations and computation time are shown in Table 4.1.

![Figure 4.2](image)

(a) (b) (c) (d)

**Figure 4.2:** A random image pair, from the same recording as Figure 4.1, corresponding Farnebäck motion vector fields estimated with resolution (256 × 136) (a), (512 × 272) (b), (1024 × 544) (c), and (2048 × 1088) (d).

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Iterations</th>
<th>Poly N</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(256 × 136)</td>
<td>15</td>
<td>5</td>
<td>9.65</td>
</tr>
<tr>
<td>(512 × 272)</td>
<td>15</td>
<td>5</td>
<td>40.07</td>
</tr>
<tr>
<td>(1024 × 544)</td>
<td>15</td>
<td>5</td>
<td>166.65</td>
</tr>
<tr>
<td>(2048 × 1088)</td>
<td>15</td>
<td>5</td>
<td>689.74</td>
</tr>
</tbody>
</table>

The results in Figure 4.2 show that increased input resolution gives a higher resolution motion vector field and a decrease in noise. Figures (b)-(d) have distinguished some or most of the edges of the moving object. The CPU time presented in Table 4.1 show a pattern where the execution time increases with the input resolution. Further results of the method are presented with resolutions (512 × 272) and (1024 × 544).

The two adjustable parameters of the method were also subjects of experimentation. First, with varying number of iterations between 5 and 20 with a polynomial
order of 5, and after that the same iterations with polynomial order 7. The results of these runs are shown in Figure 4.3, where the left column shows the simulations with polynomial order 5 and the right column shows the simulations with polynomial order 7. The corresponding Table 4.2 shows the resulting calculation time. All simulations are made with the same input as earlier with resolution \((1024 \times 544)\).

**Figure 4.3:** A random \((1024 \times 544)\)-image pair, from the same recording as Figure 4.1, corresponding Farnebäck motion vector fields estimated with varying number of iterations using polynomial order 5 (a)-(d) and 7 (e)-(h).
4.1 N-Point Fast Fourier Transform

Table 4.2: Configurations and calculation time corresponding to figures (a)-(h) in Figure 4.3.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Iterations</th>
<th>Poly N</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1024 × 544)</td>
<td>5</td>
<td>5</td>
<td>166.99</td>
</tr>
<tr>
<td>(1024 × 544)</td>
<td>10</td>
<td>5</td>
<td>166.99</td>
</tr>
<tr>
<td>(1024 × 544)</td>
<td>15</td>
<td>5</td>
<td>166.65</td>
</tr>
<tr>
<td>(1024 × 544)</td>
<td>20</td>
<td>5</td>
<td>167.30</td>
</tr>
<tr>
<td>(1024 × 544)</td>
<td>5</td>
<td>7</td>
<td>211.01</td>
</tr>
<tr>
<td>(1024 × 544)</td>
<td>10</td>
<td>7</td>
<td>210.44</td>
</tr>
<tr>
<td>(1024 × 544)</td>
<td>15</td>
<td>7</td>
<td>210.57</td>
</tr>
<tr>
<td>(1024 × 544)</td>
<td>20</td>
<td>7</td>
<td>211.65</td>
</tr>
</tbody>
</table>

The results in Figure 4.3 show a slight difference in smoothness between the choices of polynomial order for all iterations, where order 7, figures (e)-(h) show more distinct areas of motion while order 5, figures (a)-(d), show smoother transition. The figure also shows that an increased number of iterations gives some noise reduction. In figures (a) and (e), where the number of iterations is set to 5, the subject’s motion is almost unrecognizable compared to the other results. Table 4.2 shows no clear or noticeable execution time changes with the increasing number of iterations. However, the table presents an increase in execution time when the polynomial order is increased. Figure 4.4 and Table 4.3 show the result from the same simulations with resolution (512 × 272).
Figure 4.4: A random (512×272)-image pair, from the same recording as Figure 4.1, corresponding Farnebäck motion vector fields estimated with varying number of iterations using polynomial order 5 (a)-(d) and polynomial order 7 (e)-(h).
Table 4.3: Configurations and calculation time corresponding to figures (a)-(h) in Figure 4.4.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Iterations</th>
<th>Poly N</th>
<th>Calc. time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(512 × 272)</td>
<td>5</td>
<td>5</td>
<td>40.15</td>
</tr>
<tr>
<td>(512 × 272)</td>
<td>10</td>
<td>5</td>
<td>39.94</td>
</tr>
<tr>
<td>(512 × 272)</td>
<td>15</td>
<td>5</td>
<td>40.03</td>
</tr>
<tr>
<td>(512 × 272)</td>
<td>20</td>
<td>5</td>
<td>40.28</td>
</tr>
<tr>
<td>(512 × 272)</td>
<td>5</td>
<td>7</td>
<td>50.55</td>
</tr>
<tr>
<td>(512 × 272)</td>
<td>10</td>
<td>7</td>
<td>50.52</td>
</tr>
<tr>
<td>(512 × 272)</td>
<td>15</td>
<td>7</td>
<td>50.43</td>
</tr>
<tr>
<td>(512 × 272)</td>
<td>20</td>
<td>7</td>
<td>50.69</td>
</tr>
</tbody>
</table>

The results of Figure 4.4 show the same patterns as for the results with higher resolution, but now with more visible surrounding noise. The scores presented in Table 4.3 show the same pattern but with an execution time of around one-fourth of the scores from the result with higher resolution.
Horn-Schunk

As in the previous section, the Horn-Schunk method was performed with the three key points in mind, with input from the data with resolution (512 × 272), (1024 × 544), and (2048 × 1088) where remaining configurations are set as default. Parameters set for a run are the number of iterations and the positive scalar $\alpha$ mentioned in 2.3.1, where the default settings are $\text{Iterations} = 30$ and $\alpha = 1.0$. The results for resolution are shown in Figure 4.5, where the figures of the motion vector fields show a brighter nuance where the pixel movement is higher and black where the pixel is still. The configurations and computation time are shown in Table 4.4.

![Figure 4.5](image)

**Figure 4.5:** A random image pair from the data set shown in Figure 4.1 and corresponding Horn-Schunk motion vector fields estimated with resolution (256 × 136) (a), (512 × 272) (b), (1024 × 544) (c), and (2048 × 1088) (d).

**Table 4.4:** Configurations and calculation time corresponding to figures (a)-(d) in Figure 4.2.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Iterations</th>
<th>$\alpha$</th>
<th>CPU time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>(256 × 136)</td>
<td>30</td>
<td>1.0</td>
<td>18.64</td>
</tr>
<tr>
<td>(512 × 272)</td>
<td>30</td>
<td>1.0</td>
<td>85.33</td>
</tr>
<tr>
<td>(1024 × 544)</td>
<td>30</td>
<td>1.0</td>
<td>456.52</td>
</tr>
<tr>
<td>(2048 × 1088)</td>
<td>30</td>
<td>1.0</td>
<td>2277.26</td>
</tr>
</tbody>
</table>
The results presented in Figure 4.5 show an apparent reduction of surrounding noise with increasing resolution. The magnitude of the motion vectors of moving edges also seems to be higher with increasing resolution since the colored pattern on the subject becomes visible. The execution time score presented in Table 4.5 shows the same pattern as Table 4.1 for the Farnebäck-method but with double the execution time.

The two adjustable parameters of the method were also subjects of experimentation. First, with varying number of iterations between 5 and 30 with $\alpha = 3.0$, and afterward with a varying $1.0 < \alpha < 5.0$ and number of iterations fixed to 30. Results from the parameter simulations with the varying number of iterations are shown in Figure 4.6 and with the varying $\alpha$ in Figure 4.7 where the input resolution was $(1024 \times 544)$. The corresponding tables 4.5 and 4.6 show the resulting calculation times where the simulations are made with the same input as earlier simulations.

![Figure 4.6: A random image pair from the data set shown in Figure 4.1 and corresponding Horn-Schunk motion vector fields estimated with 5 (a), 10 (b), 20 (c), and 30 (d) number of iterations.](image)

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Iterations</th>
<th>$\alpha$</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(1024 \times 544)$</td>
<td>5</td>
<td>3.0</td>
<td>105.99</td>
</tr>
<tr>
<td>$(1024 \times 544)$</td>
<td>10</td>
<td>3.0</td>
<td>180.28</td>
</tr>
<tr>
<td>$(1024 \times 544)$</td>
<td>20</td>
<td>3.0</td>
<td>328.11</td>
</tr>
<tr>
<td>$(1024 \times 544)$</td>
<td>30</td>
<td>3.0</td>
<td>475.08</td>
</tr>
</tbody>
</table>
The result of Figure 4.6 shows no real difference with the increasing number of iterations, while the score in Table 4.5, different from the score from Table 4.3, shows an increasing execution time for the higher number of iterations.

![Figure 4.7: A random image pair from the data set shown in Figure 4.1 and corresponding Horn-Schunk motion vector fields estimated with $\alpha = 1.0$ (a), $\alpha = 1.0$ (b), $\alpha = 1.0$ (c), and $\alpha = 1.0$ (d).](image)

Table 4.6: Configurations and calculation time corresponding to figures (a)-(d) in 4.7

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Iterations</th>
<th>$\alpha$</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1024 × 544)</td>
<td>5</td>
<td>1.0</td>
<td>105.55</td>
</tr>
<tr>
<td>(1024 × 544)</td>
<td>5</td>
<td>2.0</td>
<td>104.53</td>
</tr>
<tr>
<td>(1024 × 544)</td>
<td>5</td>
<td>3.0</td>
<td>105.99</td>
</tr>
<tr>
<td>(1024 × 544)</td>
<td>5</td>
<td>5.0</td>
<td>104.11</td>
</tr>
</tbody>
</table>

The results of Figure 4.7 show more detailed movement at edges when $\alpha$ increases, together with some noise reduction. There are no visible patterns or changes in the execution time presented in Table 4.6.
4.1 N-Point Fast Fourier Transform

4.1.2 Frequency result

This section presents the frequency results from the N-point FFT method using the data set from Section 3.1. The results include quantitative scores of estimated breathing frequency in relation to the true breathing rate and results where the estimated frequencies are displayed. The frequency result for some essential configuration differences will be shown, but the final configurations for the Farnebäck-estimated motion vectors will be used for the majority of the results. Section 5.1 describes how these configurations were selected. Finally, the frequency result of different image sequences will be presented for different input image resolutions, and two different types of optronic devices.

Visual differences

The data set used during the project has many subjects (cows) with different appearances. The nuance and color of the hair can differ a lot, as well as the pattern of spots. This section presents the results of three simulations of cows with different visual appearances, shown in Figure 4.8.

(a)  (b)  (c)

Figure 4.8: Three images from the data set with visual differences

The results are presented with three figures types, one with the spatial location of all estimated frequencies in a specific frame, where blue boxes indicate the locations of estimated breathing frequencies and red boxes indicate found frequencies discarded as breathing. This figure type is presented in the top row of Figure 4.9. The second figure type only shows the locations of the estimated breathing frequency in green, presented in the bottom row of Figure 4.9. The third figure type, presented in Figure 4.10, shows the estimated frequency over a one-minute interval, additional to the initial 256 frames used to calculate the first estimation. Blue markings denote an estimated breathing frequency in beats per minute (BPM), and red markings an estimated breathing frequency calculated with an error covariance too large for the estimation to be accepted. Table 4.7 presents the score of the three simulations together with the true breathing rate and a calculated error margin.
Figure 4.9: Spatial locations for all estimated frequencies (red and blue) in the top row. Spatial locations for estimated breathing frequencies (green) in the bottom row. (a), (b) and (c) corresponds to (a), (b) and (c) from Figure 4.8.
From the results in Figure 4.9, there are some indications that the estimated breathing frequency is located on the edges of the cow or along the colored spots in the cows’ fur. There is also an indication that frames of more bright-haired cows have a higher occurrence of breathing frequency. There are no real patterns regarding the frequency noise located outside the cow.

**Figure 4.10:** The estimated frequency over a one minute interval where (a), (b) and (c) corresponds to (a), (b) and (c) from Figure 4.9.
Table 4.7: Configurations and calculation times corresponding to figures (a)-(c) in Figure 4.10

<table>
<thead>
<tr>
<th>Cam-Rec</th>
<th>Resolution</th>
<th>True BPM</th>
<th>est. BPM</th>
<th>ε</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basler 7</td>
<td>(512 × 272)</td>
<td>24.75</td>
<td>23.37</td>
<td>-1.38</td>
</tr>
<tr>
<td>Basler 10</td>
<td>(512 × 272)</td>
<td>27.75</td>
<td>26.40</td>
<td>-1.35</td>
</tr>
<tr>
<td>Basler 11</td>
<td>(512 × 272)</td>
<td>27.75</td>
<td>26.49</td>
<td>-1.26</td>
</tr>
</tbody>
</table>

The results of the estimated breathing frequency over one minute presented in Figure 4.10 show that simulations (b) and (c) keep a relatively continuous shape during the whole minute in the region of the true BPM presented in Table 4.7. Simulation (a) differs between 50 and 60 seconds. The mean estimated breathing frequency score in the table shows that the two cows with colored spots, (b) and (c), score a result closer to the true breathing rate. There is also a pattern that all estimations are lower than the true frequencies. The same simulations but with Horn-Schunk motion estimation show the same pattern but at a lower performance level.
RealSense results

The data set also includes some recordings with the RealSense camera, as mentioned in Section 3.1. These recordings are set at different locations in relation to the Basler recordings and also differ some in resolution and frame rate. Here, the results of two simulations with RealSense recordings from two different locations are presented in the same way as for the Basler recordings. One figure with the spatial location of all estimated frequencies in a specific frame, where blue boxes indicate the locations of estimated breathing frequencies and red boxes indicate found frequencies discarded as breathing. This figure type is presented in the top row of Figure 4.9. The second figure type only shows the locations of the estimated breathing frequency in green, presented in the bottom row of Figure 4.9. The third type of figure, presented in Figure 4.12, shows the estimated frequency over a one-minute interval, additional to the initial 256 frames used to calculate the first estimation. Blue markings denote an estimated breathing frequency in beats per minute (BPM), and red markings an estimated breathing frequency calculated with an error covariance too large for the estimation to be accepted. Table 4.8 presents the score of the two simulations together with the true breathing rate and a calculated error margin.

Figure 4.11: Spatial locations for all estimated frequencies (red and blue) in the top row. Spatial locations for estimated breathing frequencies (green) in the bottom row. Images from three different recordings done with the RealSense camera.

From the results in Figure 4.11, it can be seen that there are clearly frequencies in the frame not coming from the cow whose breathing frequency is estimated. In figures (a) and (b), there are other cows in the background producing noise, and in figure (c), there seems to be more noise than in previous results from the Basler recordings. The green boxes indicating the location of the breathing frequency are found on the edges of the subjects for (a) and (c), like in earlier results, and on the whole back half of the subject in (b).
Figure 4.12: The estimated frequency over a one minute interval where (a), (b) and (c) corresponds to (a), (b) and (c) from Figure 4.11.
Table 4.8: Configurations and estimated frequency corresponding to figures (a)-(c) in Figure 4.12

<table>
<thead>
<tr>
<th>Cam-Rec</th>
<th>Resolution</th>
<th>True BPM</th>
<th>est. BPM</th>
<th>$\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RealSense 36</td>
<td>(640 × 360)</td>
<td>33.50</td>
<td>31.58</td>
<td>-1.92</td>
</tr>
<tr>
<td>RealSense 10</td>
<td>(640 × 360)</td>
<td>32.00</td>
<td>42.88</td>
<td>10.88</td>
</tr>
<tr>
<td>RealSense 28</td>
<td>(640 × 360)</td>
<td>37.50</td>
<td>44.50</td>
<td>7.00</td>
</tr>
</tbody>
</table>

The results of the estimated breathing frequency over one minute presented in Figure 4.12 shows jagged plots for (a) and (b) where the estimation jumps between some different frequencies. The result for (c) is the most continuous of the three, and as mentioned before, is the only recording with only one cow in the frame. The mean of the estimated breathing frequency score presented in Table 4.8 show bad estimations for (a), (b), and (c) with higher error margin compared to earlier simulations.

Results of Varying the Image Resolution

After the results of motion vectors with varying resolution for the input images presented in Section 4.1.1, some patterns and information about the behavior were noted. Here, the results of three simulations of input images with different resolutions are presented. Since the differing resolution changes the number of pixels in the image, the (16 × 16)-boxes mentioned in Section 3.3 are scaled with the different resolutions. The three resolutions are (256 × 136), (512 × 272) and, (1024 × 544), where the size of the corresponding "boxes" are (8 × 8), (16 × 16), and (32 × 32). The results are presented with three figures, one with the spatial location of all estimated frequencies in a specific frame, where blue boxes indicate the locations of estimated breathing frequencies and red boxes indicate found frequencies discarded as breathing. These figure types are presented in the top row of Figure 4.9 where all figures represent the same frame at $t = 25$ s. The second figure type only shows the locations of the estimated breathing frequency in green, presented in the bottom row of Figure 4.9 where all figures represent the same frame at $t = 25$ s. The third figure type, presented in Figure 4.14, show the estimated frequency over a one-minute interval, additional to the initial 256 frames used to calculate the first estimation. Blue markings denote an estimated breathing frequency in beats per minute (BPM), and red markings an estimated breathing frequency calculated with an error covariance too large for the estimation to be accepted. Table 4.9 presents the score of the three simulations together with the true breathing rate, a calculated margin of error, and the mean execution time for the whole process.
Figure 4.13: Spatial locations for all estimated frequencies (red and blue) in the top row. Spatial locations for estimated breathing frequencies (green) in the bottom row. Images from Basler recording 07 with three different resolutions at time $t = 25$ s.
From the results in Figure 4.13, there is a pattern where increased resolution results in more frequencies, as shown in the top line of images. The estimated breathing frequency is seemingly found in the same location for all figures (a)-(c), but with a higher amount of green boxes when resolution is increased.

Figure 4.14: The estimated frequency over a one minute interval where (a), (b) and (c) corresponds to (a), (b) and (c) from Figure 4.13
Table 4.9: Configurations, estimated breathing frequencies and calculation time corresponding to figures in 4.14

<table>
<thead>
<tr>
<th>Cam-Rec</th>
<th>Resolution</th>
<th>True BPM</th>
<th>est. BPM</th>
<th>$\epsilon$</th>
<th>CPU time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basler 15</td>
<td>(256 × 136)</td>
<td>35.5</td>
<td>34.78</td>
<td>-0.72</td>
<td>34.33</td>
</tr>
<tr>
<td>Basler 15</td>
<td>(512 × 272)</td>
<td>35.5</td>
<td>34.32</td>
<td>-1.18</td>
<td>68.37</td>
</tr>
<tr>
<td>Basler 15</td>
<td>(1024 × 544)</td>
<td>35.5</td>
<td>35.54</td>
<td>0.04</td>
<td>208.66</td>
</tr>
</tbody>
</table>

The results of the estimated breathing frequency over one minute presented in Figure 4.14 show that the behavior does not change drastically with the different resolutions. All three results react to some disturbance between $t = 10$ and $t = 20$ but later on stays around the margins of the true BPM. The mean estimated breathing frequency score in the Table 4.9 show that (a) and (b) have estimations with around 1 BPM difference to the true BPM while (c) is much closer, though the execution time is about 4 times longer.

Chosen Configuration

The same configuration was used on the entirety of the data set, i.e Farnebäck-estimated motion vectors with optimal settings and input images with resolution (512 × 272) from the Basler-camera and (640x360) from the RealSense-camera. The results are presented in the same way i.e, Table 4.13 show the mean estimated breathing frequency from each simulation together with the true breathing rate and a calculated margin of error. Due to the number of simulations, no results are presented here.

4.2 Breathing Movement Zero-Crossing Counting

The following section presents the results from the Breathing Movement Zero-Crossing Counting-method, described in Section 3.2, when simulated with different sets of parameters for the Lucas-Kanade algorithm. Next, the simulations with varying parameters regarding the optical flow are presented in Section 4.2.1 and Section 4.2.2 presents the simulations with varying image resolution. Finally, Section 4.2.3 provides the chosen configuration of parameters used when estimating the respiratory rate on the entire data set.

4.2.1 Motion Vectors with Varying Parameters

This section presents the results of the calculated motion vectors with corresponding calculation time when simulating with different box sizes and number of pixels to track in the Lucas-Kanade method from Section 3.2.2. The simulations were done on randomly selected image pairs from multiple recordings.
Optimizing the Box Size

The simulations for optimizing the box size, which corresponds to the number of neighboring pixels, from Section 3.2.2, were executed with four different box sizes on image pairs with resolution (1024 × 544). Figure 4.15 presents the simulations from the Basler 11 recording, where the green and red arrows correspond to motion vectors with positive and negative horizontal velocities, respectively. The blue square is the ROI tracker from Section 3.2.1.

![Figure 4.15: Motion vectors calculated with different box sizes for the Basler 11 recording.](image)

Table 4.10: Calculation time for different box sizes corresponding to figures (a)-(d) in Figure 4.15

<table>
<thead>
<tr>
<th>Box size</th>
<th>CPU time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>(25 × 25)</td>
<td>21.13</td>
</tr>
<tr>
<td>(50 × 50)</td>
<td>27.27</td>
</tr>
<tr>
<td>(75 × 75)</td>
<td>37.63</td>
</tr>
<tr>
<td>(100 × 100)</td>
<td>51.73</td>
</tr>
</tbody>
</table>

Noise cancellation is the main objective when selecting the box size and a noisy optical flow can be observed when either the red and green arrows are mixed in an unclear pattern or when motion vectors occur on static objects, like the metal frame surrounding the cow. For example, the simulation in figure (a) yields a very noisy optical flow where the green and red arrows are heavily mixed. On
the other hand, simulations in figures (b)-(d) display a similar optical flow with slightly more noise in simulation (b), which can be seen on the circular metal object in the lower-left corner of the ROI tracker. Finally, the calculation times presented in Table 4.10 indicate a significant increase in calculation time with box size.

**Optimizing the Number of Pixels to Track**

Simulations were done to optimize the number of pixels to track, which relates to the density of the optical flow in the Lucas-Kanade method from Section 3.2.2. The presented simulations were performed on the same image pair from the Basler 11 recording with four different image resolutions to detect deviations between resolutions visually. The aim is to cover the cow with as few pixels as possible to minimize calculation time and maintain high enough performance.

The calculated optical flow of the simulations are presented in Figure 4.16 with 576, 1089, and 5184 tracked pixels in row one, two, and three for image resolution (256 × 136), (512 × 272), (1024 × 544), and (2048 × 1088) in column one, two, three and four. All simulations were performed with a box size set as (75 × 75)

![Figure 4.16: Calculated optical flow with 576, 1089 and 5184 tracked pixels in row one, two and three with image resolution (256 × 136), (512 × 272), (1024 × 544) and (2048 × 1088) in column one, two, three and four.](image-url)
Table 4.11: Calculation time of the optical flow with different number of motion vectors and image resolution corresponding to figures in Figure 4.16

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Pixels to track</th>
<th>CPU time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>(256 × 136)</td>
<td>576</td>
<td>10.73</td>
</tr>
<tr>
<td>(512 × 272)</td>
<td></td>
<td>17.10</td>
</tr>
<tr>
<td>(1024 × 544)</td>
<td></td>
<td>26.49</td>
</tr>
<tr>
<td>(2048 × 1088)</td>
<td></td>
<td>33.98</td>
</tr>
<tr>
<td>(256 × 136)</td>
<td>1089</td>
<td>20.14</td>
</tr>
<tr>
<td>(512 × 272)</td>
<td></td>
<td>31.56</td>
</tr>
<tr>
<td>(1024 × 544)</td>
<td></td>
<td>51.36</td>
</tr>
<tr>
<td>(2048 × 1088)</td>
<td></td>
<td>62.17</td>
</tr>
<tr>
<td>(256 × 136)</td>
<td>5184</td>
<td>91.65</td>
</tr>
<tr>
<td>(512 × 272)</td>
<td></td>
<td>135.23</td>
</tr>
<tr>
<td>(1024 × 544)</td>
<td></td>
<td>194.17</td>
</tr>
<tr>
<td>(2048 × 1088)</td>
<td></td>
<td>420.25</td>
</tr>
</tbody>
</table>

What can be observed overall is that the optical flow gradually changes for every enhancement of the image resolution. For example, in columns one and two, the motion vectors indicate that the bottom-right part of the cow is moving to the right. But, in columns three and four, the movement is opposite for the same part of the cow. Furthermore, the simulations in row three show a relatively dense optical flow, especially for resolution (256 × 136) where the motion vectors tend almost to overlap each other and therefore calculate the optical flow in the same small area multiple times. Meanwhile, the two upper-most rows present a more sparse optical flow but with the same characteristics as their respective simulation with 5184 tracked pixels. However, the calculation time increases significantly with the number of pixels to track and image resolution.
4.2.2 Estimated Frequency with Varying Image Resolution

This section presents the estimated respiratory rate and the signals and filter parameters used in the zero-crossings algorithm from Sections 3.2.3-3.2.6 when simulating with different image resolutions. Simulations were performed with the Basler11 and Basler 07 recordings to detect variations in the results for different image resolutions. The resolutions tested were $(256 \times 136)$, $(512 \times 272)$, $(1024 \times 544)$ and $(2048 \times 1088)$. Figure 4.17 displays still frames of the recordings with cell division overlaid.

Figure 4.17: Still frame with cell division overlaid for the Basler 11 and Basler 07 recordings.

Extracting Breathing Movements

Figure 4.18 visualizes the results estimated in a cell using the methodology in Section 3.2.3 applied on the two separate cow videos, also using varying image resolutions. The simulations were done on a 20-second interval with 1089 pixels to track and a box size equal to $(75 \times 75)$. The median velocities $u(t)_{i,j}$ and $y(t)_{i,j}$ are represented with the blue and magenta curves, respectively. The noise-canceled estimated breathing movement, $y(t)_{i,j}^{BP}$, is represented with the black curve.
Figure 4.18: The black line represents the signal $y(t)_{i,j}^{BP}$, which is the extracted and noise canceled breathing movements. The blue and magenta lines are the median velocities $u(t)_{i,j}$ and $y(t)_{i,j}$, respectively.
Figure (a)-(d) represents the Basler 11 recording, a relatively periodic noise canceled signal can be observed over the interval. However, the simulation in figure (a) differs compared to (b)-(d), especially for the negative velocities. This problem was shown previously in the simulations in Section 4.2.1, where the calculated optical flow wasn’t consistent for every image resolution. One noticeable feature in the estimated breathing movement is that; the intake and exhaust differ, as mentioned in Section 3.2.3. The exhaust seems to be executed with one rapid out blow while the intake is a process with stable velocity for a more extended time, leading to some ripples in the black signal in the figure. Figures (e)-(h) present the simulations for recording 07 from the Basler camera. The velocities of these signals are significantly smaller and less periodic than for the Basler 11 recording. One reason for this is that the cow is black-haired with no patterns. Therefore, the optical flow calculations could be relatively inaccurate. However, unlike (a)-(d), the computed signals seem to reduce in velocity and become more periodic with higher image resolution.

**Parameter Estimation and Filtering**

The filter parameters for the bandpass filter, $h_{2\text{BP}}$, are estimated with the power spectrums in Figure 4.19. Figures (a)-(h) represent the signals presented in figures (a)-(h) in 4.18, respectively. The most and second most distinguished frequency components, $f_1$ and $f_2$, are presented in red and blue. The interval $[f_1 - 1, f_1 + 1]$ is excluded with finding $f_2$. The quality measure, $Q$, is presented for every image resolution and approaches one the more the amplitude at $f_1$ differs from the amplitude at $f_2$.

As observed in Figure 4.18, the extracted breathing movements for cow 11 are more periodic than for cow 07. This can also be seen by studying the quality measurements of the power spectrums, where the quality $Q$ for cow 11 outperforms the quality for cow 07 for every image resolution. However, figures (e)-(h) show that the signal becomes more periodic with higher image resolution, as mentioned in the previous section.

Figure 4.20 presents the final estimated signal, $\hat{y}(t)_{i,j}$, in green for every image resolution. This signal corresponds to the signal $y(t)_{i,j}^{BP}$ passed through the bandpass filter, $h_{2\text{BP}}$, with calculated filter parameters from the power spectrums.
Figure 4.19: Power spectrum of the extracted breathing movements with different image resolutions for cow 11 and cow 07 recorded with the Basler camera.
Figure 4.20: The green lines are the bandpass filtered breathing movements in a cell, calculated with different image resolutions for cows in Basler recording 11 and 07.
BPM Estimation

The zero-crossings algorithm presented in Section 3.2.6 is applied to the signal, $\hat{y}(t)_{i,j}$, from Figure 4.20 for every image resolution. The zero-crossings are visualized in blue in Figure 4.21. Finally, the estimated BPM is presented for every image resolution and cow in Table 4.12 with its corresponding quality measure, $Q$, estimation error, $\epsilon$, and calculation time.

For cow 11, the simulation with resolution $(512 \times 272)$ and $(1024 \times 544)$ estimate the BPM accurately with an error of 0.27 BPM. But, the estimated respiratory rate differs by almost 2.5 BPM between the lowest and highest resolution. These results are obtained with a quality measure between 0.74 and 0.80. The results for Cow 07 display fewer variations in the estimated BPM. The lowest resolution yields the worst result and, therefore, a significantly larger estimation error compared to the higher resolutions. The quality of the estimations varies between 0.16 and 0.54, which is significantly lower than the qualities for cow 11. However, these quality measurements are not directly related to the estimation error but hint that cow 07 is a more challenging object to observe than cow 11. The calculation time for all simulations indicates a significant increase in calculation time with image resolution.

**Table 4.12**: Estimated BPM, estimation error, quality and calculation time from the zero-crossings in Figure 4.21 for a 20 second interval with different image resolutions for cow 11 and cow 07.

<table>
<thead>
<tr>
<th>Cam rec</th>
<th>Resolution</th>
<th>True BPM</th>
<th>est. BPM</th>
<th>$\epsilon$</th>
<th>$Q$</th>
<th>CPU time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basler 11</td>
<td>$(256 \times 136)$</td>
<td>27</td>
<td>26.08</td>
<td>-0.92</td>
<td>0.80</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>$(512 \times 272)$</td>
<td></td>
<td>27.27</td>
<td>0.27</td>
<td>0.76</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>$(1024 \times 544)$</td>
<td></td>
<td>27.27</td>
<td>0.27</td>
<td>0.78</td>
<td>170</td>
</tr>
<tr>
<td></td>
<td>$(2048 \times 1088)$</td>
<td></td>
<td>28.57</td>
<td>1.57</td>
<td>0.74</td>
<td>440</td>
</tr>
<tr>
<td>Basler 07</td>
<td>$(256 \times 136)$</td>
<td>25.5</td>
<td>24.00</td>
<td>-1.5</td>
<td>0.16</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>$(512 \times 272)$</td>
<td></td>
<td>25.00</td>
<td>-0.5</td>
<td>0.44</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>$(1024 \times 544)$</td>
<td></td>
<td>25.00</td>
<td>-0.5</td>
<td>0.47</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>$(2048 \times 1088)$</td>
<td></td>
<td>25.00</td>
<td>-0.5</td>
<td>0.54</td>
<td>410</td>
</tr>
</tbody>
</table>
Figure 4.21: Calculated zero-crossings with different image resolutions for cow 11 and cow 07.

(a) Basler11, Res: (256 × 136)  
(b) Basler11, Res: (512 × 272)  
(c) Basler11, Res: (1024 × 544)  
(d) Basler11, Res: (2048 × 1088)  
(e) Basler07, Res: (256 × 136)  
(f) Basler07, Res: (512 × 272)  
(g) Basler07, Res: (1024 × 544)  
(h) Basler07, Res: (2048 × 1088)
4.2.3 Chosen Parameter Configuration

One parameter configuration was chosen when estimating the respiratory rate on the data set recorded with the Basler camera at the milking robot. The simulations from Sections 4.2.1 and 4.2.2 resulted in selecting the box size (75x75) while tracking 1089 pixels with a (512 x 272) image resolution. These choices are motivated in the discussion in Chapter 5.
4.3 Comparing Estimated BPM for the N-point FFT Method and the Zero-crossing Method

This section presents the results of simulations done with the two methods’ final configurations. The simulations are done with all Basler recordings from the milking machine, i.e., all Basler recordings where a true BPM could be determined. Table 4.13 presents the true BPM in relation to the estimated BPM for each method together with the mean error.

Table 4.13: Estimated BPM and estimation error for the N-point FFT method and zero-crossings method.

<table>
<thead>
<tr>
<th>Cam rec</th>
<th>Method</th>
<th>True BPM</th>
<th>est. BPM</th>
<th>ϵ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basler 02</td>
<td>N-Point FFT</td>
<td>30.75</td>
<td>28.15</td>
<td>-2.60</td>
</tr>
<tr>
<td></td>
<td>Zero-Crossing</td>
<td></td>
<td>30.00</td>
<td>-0.75</td>
</tr>
<tr>
<td>Basler 05</td>
<td>N-Point FFT</td>
<td>28.00</td>
<td>22.80</td>
<td>-5.20</td>
</tr>
<tr>
<td></td>
<td>Zero-Crossing</td>
<td></td>
<td>27.27</td>
<td>-0.73</td>
</tr>
<tr>
<td>Basler 07</td>
<td>N-Point FFT</td>
<td>24.75</td>
<td>23.37</td>
<td>-1.38</td>
</tr>
<tr>
<td></td>
<td>Zero-Crossing</td>
<td></td>
<td>25.00</td>
<td>0.25</td>
</tr>
<tr>
<td>Basler 08</td>
<td>N-Point FFT</td>
<td>36.75</td>
<td>36.31</td>
<td>-0.44</td>
</tr>
<tr>
<td></td>
<td>Zero-Crossing</td>
<td></td>
<td>37.50</td>
<td>0.75</td>
</tr>
<tr>
<td>Basler 10</td>
<td>N-Point FFT</td>
<td>27.75</td>
<td>26.40</td>
<td>-1.35</td>
</tr>
<tr>
<td></td>
<td>Zero-Crossing</td>
<td></td>
<td>27.27</td>
<td>-0.48</td>
</tr>
<tr>
<td>Basler 11</td>
<td>N-Point FFT</td>
<td>27.75</td>
<td>26.49</td>
<td>-1.26</td>
</tr>
<tr>
<td></td>
<td>Zero-Crossing</td>
<td></td>
<td>27.27</td>
<td>-0.48</td>
</tr>
<tr>
<td>Basler 12</td>
<td>N-Point FFT</td>
<td>33.50</td>
<td>35.82</td>
<td>2.32</td>
</tr>
<tr>
<td></td>
<td>Zero-Crossing</td>
<td></td>
<td>37.50</td>
<td>4.00</td>
</tr>
<tr>
<td>Basler 13</td>
<td>N-Point FFT</td>
<td>24.00</td>
<td>18.01</td>
<td>-5.99</td>
</tr>
<tr>
<td></td>
<td>Zero-Crossing</td>
<td></td>
<td>23.08</td>
<td>-0.92</td>
</tr>
<tr>
<td>Basler 14</td>
<td>N-Point FFT</td>
<td>27.50</td>
<td>26.68</td>
<td>-0.82</td>
</tr>
<tr>
<td></td>
<td>Zero-Crossing</td>
<td></td>
<td>27.27</td>
<td>-0.23</td>
</tr>
<tr>
<td>Basler 15</td>
<td>N-Point FFT</td>
<td>35.50</td>
<td>34.32</td>
<td>-1.18</td>
</tr>
<tr>
<td></td>
<td>Zero-Crossing</td>
<td></td>
<td>35.29</td>
<td>-0.21</td>
</tr>
<tr>
<td>Basler 17</td>
<td>N-Point FFT</td>
<td>30.75</td>
<td>28.00</td>
<td>2.75</td>
</tr>
<tr>
<td></td>
<td>Zero-Crossing</td>
<td></td>
<td>31.58</td>
<td>0.83</td>
</tr>
<tr>
<td>Basler 18</td>
<td>N-Point FFT</td>
<td>25.50</td>
<td>22.84</td>
<td>-2.66</td>
</tr>
<tr>
<td></td>
<td>Zero-Crossing</td>
<td></td>
<td>26.09</td>
<td>0.59</td>
</tr>
<tr>
<td>Basler 19</td>
<td>N-Point FFT</td>
<td>26.50</td>
<td>23.97</td>
<td>-2.53</td>
</tr>
<tr>
<td></td>
<td>Zero-Crossing</td>
<td></td>
<td>27.27</td>
<td>0.77</td>
</tr>
<tr>
<td>RealSense 10</td>
<td>N-Point FFT</td>
<td>32.00</td>
<td>42.88</td>
<td>10.88</td>
</tr>
<tr>
<td></td>
<td>Zero-Crossing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RealSense 28</td>
<td>N-Point FFT</td>
<td>27.50</td>
<td>44.50</td>
<td>7.00</td>
</tr>
<tr>
<td></td>
<td>Zero-Crossing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RealSense 36</td>
<td>N-Point FFT</td>
<td>33.50</td>
<td>31.58</td>
<td>-1.92</td>
</tr>
<tr>
<td></td>
<td>Zero-Crossing</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This chapter ties the methods presented in Chapter 3 and results presented in Chapter 4 together with discussions.

5.1 Motion vectors for N-point FFT

In Section 4.1.1, the Farnebäck method with input resolution \((512 \times 242)\) and 20 iterations results in almost as good results as with higher input resolution. The motion vector noise around the subject was higher with resolution \((256 \times 121)\), and since the execution time is low for both, \((512 \times 242)\) was selected as the preferred input resolution. Since the number of iterations doesn’t show any noticeable increase in execution time, as the input resolution does, it is the best combination of input and configuration for Farnebäck’s motion vector estimation. For the whole system to perform in real-time, its execution time needs to be less than once second if we require that the system produces frequency update one per second. With the assumption that the breathing frequency can reach over 60 bpm, and since the system needs several samples per breath, the execution time for the motion vectors as chosen to be limited to \(CPU_{\text{time}} < 200 \text{ [ms]}\). The results of the motion vectors estimated with the Horn-Schunk algorithm configurations \(\text{Iterations} = 5\) with input image resolution \((1024 \times 544)\) are within that limit. However, these configurations only give good results for cows with distinct hair color patterns, and therefore do not work as intended for monochromatic cows. There are a lot of minor tests that are not included in this thesis, and there can be more experimentation done to optimize the choice of algorithm and parameter configuration. With the work hours and available and the generally good result in terms of the three key points presented in Section 4.1.1, the Farnebäck algorithm with parameters \(\text{Iterations} = 20\) and \(\text{PolyN} = 7\) was selected as the
preferred method and configuration for the N-point FFT-estimation of breathing frequency.

5.2 Frequency Estimation with N-point FFT

The implementation of the N-point FFT method presented in Section 3.3.2 with results from Section 4.1.2 shows the reaction of the methods to cows of different colors and sizes. As mentioned in the result, the magnitude of the maximum frequency found in the estimation is higher when there are distinct spots or patterns in the subject’s fur, which makes the system less likely to change frequency quickly, i.e react to sudden noise. One possible solution to the problem is to fix a brighter background. Since the grayscale pixel values of the ground are close to the darker parts of some cows, the motion vector estimation has problem finding what pixels are cow and which are background. A more distinct color difference would make the edges of the abdomen more prominent and hopefully thereby give a higher amount of estimated breathing frequencies. However, it needs to be done before the recording.

When viewing the results of the recordings from the RealSense camera, the performance is significantly worse than with the Basler recordings from the milking robot. The presence of other cows and the room for the subject to move around in produces more frequencies in the frame not related to the subjects breathing. The presence itself does not interfere with the estimation of the subjects breathing but the automated solution to decide that frequency by neighborhood weighing and a Point-Mass inspired filter is designed to handle sudden or short-time noise and movements.

The input resolution was discussed in Section 5.1, and with the results from Section 4.1.2 there are no real changes in the execution time. The estimated frequency presented in Table 4.1 shows that resolution (512 × 247) preforms worst of the three, although the difference is small. After multiple simulations with different inputs from the data set, we concluded the higher resolution is more reliable, and with the execution time $CPU_{\text{time}} = 68.37 \, [\text{ms}]$ the process can be used at least ten times per second and still be within the limits of real-time estimation.

In the development of the overall process, some choices are worth discussing in terms of implementation. In Section 2.2, the Gaussian filter is introduced, which can be used as a low pass-filter for each input image. The fine limit of removing noise but not losing too much detail is the dilemma at hand. Since the detail of edges is of high importance together with the information loss in the input downscale process, a Gaussian filter was considered too much, but if the input images are chosen with higher resolution and fewer samples per second, it could be useful. The input image size also corresponds to the size of the boxes where the mean motion vectors are calculated. The choice for it to be (16 × 16) is to lower the magnitude of calculations enough while not group too much of the moving edges with the stationary background. The size can be scaled with the input with
(16 × 16) corresponding to input size (512 × 247), but it can operate with all input sizes as a standard size. Furthermore, the zero-padding process is done with double length but can be scaled up even more to increase the frequency resolution. When reviewing all results of the estimated breathing frequency over one minute presented in Section 4.1.2 together with the industrial use of a hypothetical future product as presented in Section 1.1, there is no real need to know the breathing frequency with more precision than a couple of steps between each integer measured in BPM. At last, the Neighborhood Weighing and point-mass inspired filter are tailor-made for the problem. It takes advantage of the fact that breathing movements are found at some specific areas of the subject’s body without having the part’s spatial location in the image. Using earlier measurements as a probability distribution for the current estimation also gives less reaction to sudden noise or movements, as mentioned before. Discarding estimations where the covariance is too high gives a stabilizing effect on the estimated frequency. The width of the kernel of Gaussian distribution, \( w_N \), presented in Section 3.3.4, affects how quick the system reacts to changes.

5.3 Optical Flow for Breathing Movement Zero-Crossing Counting

In Section 4.2.1, the simulations with varying box sizes showed that size (25 × 25) results in an extremely noisy optical flow and is, therefore, not an option. The remaining sizes displayed similar results where size (75 × 75) and (100 × 100) slightly outperforms size (50 × 50) when it comes to noise cancellation. Since noise cancellation was the primary goal when optimizing, size (50 × 50) was discarded. The two remaining options have no observable differences in the optical flow and, therefore, size (75 × 75) was used as the setting due to the shorter calculation time. A demand for further simulations with box sizes between (50 × 50) and (75 × 75) wasn’t performed since the noise cancellation and calculation time was satisfying enough. Instead, the remaining time was distributed to more urgent and time-consuming problems. The selected box size could be seen as relatively large compared to some image resolutions used during the simulations. However, a hypothesis derived from Section 2.3 is that the tracked pixels can be far from any easily observable edges in the frame. Therefore, a larger box has a higher probability of containing an edge.

The secondary simulations in Section 4.2.1 were done to optimizing the number of pixels to track in the Lucas-Kanade algorithm. Three arbitrary numbers of pixels were simulated, also with varying image resolutions. One main observation is that the optical flow characteristics don’t change with the number of tracked pixels, but the density does. The simulations with 5184 tracked pixels show a relatively dense optical flow. Many of the calculated motion vectors can be seen as redundant and already expressed by a neighboring motion vector. This, together with increased calculation time with the number of pixels, discarded using 5184 pixels as the setting. However, to ensure that all image resolutions maintain a
relatively dense optical flow with a reasonable calculation time, 1089 pixels to track were considered a valid option. All though, the optimal number might not be 1089 but it yielded good enough results to keep progressing the development.

5.4 Frequency Estimation with Breathing Movement

Zero-Crossing Counting

The simulations with varying image resolution presented in Section 4.2.2 resulted in a relatively similar estimation error for every resolution. However, there are some significant differences on the way to the final results that need to be addressed. These simulations were done with the cows from Basler 11 and Basler 07. They will in this section be referred to as cow 11 and cow 07.

Firstly, comparing both cows’ calculated median velocities in Figure 4.18 shows a significant difference in velocity. There are two explanations for this. The first is that cow 07 moves in a slower motion than cow 11. The second explanation is that cow 07 is monochromatic black and has no observable edges. As presented in Section 2.3.2, the Lucas-Kanade algorithm assumes that the intensity of the neighborhood surrounding the tracked pixel is constant. If this neighborhood isn’t standing out from the rest of the cow, the pixel can be tracked in the wrong direction. This will result in a noisy optical flow where the direction of the motion vectors is mixed, leading to median velocity that approaches zero. For cow 11 in Figure 4.18, figure (a) shows a significantly reduced velocity compared to (b)-(d), which indicates that some motion vectors are miscalculated. However, the signals in figure (a) are still relatively smooth and periodic. This is most likely because cow 11 is a white-haired cow with several brown-haired patches and is an excellent object to track.

Next, the power spectrums in Figure 4.19 present the detected frequency components found in the bandpass filtered signals $y(BP_{i,j})$. By studying the plots, one can get an opinion that cow 11 is far better tracked than cow 07. However, the purpose of calculating the power spectrums is to obtain the fundamental frequency from a signal with multiple frequency components. The quality measure in figure (e) is extremely low and indicates that the measurement for cow 07 with resolution (256 × 136) shouldn’t be trusted. For the monochromatic cow 07, the quality is increased with the resolution, which means that a relatively high image resolution should be considered an input to the optical flow algorithm.

Finally, after bandpass filtering, the zero-crossings algorithm was applied to the estimated signal in Figure 4.21. Since the same fundamental frequency was found for every image resolution for both cows, these signals have the same characteristics with the same amplitude difference explained earlier. This proves that a bad quality measure can estimate the true respiratory frequency but should not be trusted as much as the measurements with higher quality. The estimated respiratory frequencies are best for resolution (512 × 272) and (1024 × 544) with an estimation error at 0.27 for cow 11 and -0.5 for cow 07. This, together with the
previously presented observations and the calculation time increases with image resolution, results in selecting (512 × 272) as the preferred image resolution.

5.5 General Discussion

When studying the results of Table 4.13, it is noticeable that the Breathing Movement Zero-Crossing Counting algorithm performs better than the N-point FFT algorithm in terms of margin error. It also seems to be the more consistent algorithm, with only one margin error |e| > 0.92, compared to the N-point FFT algorithm, which estimated breathing frequencies with margin errors between 0.44 ≤ |e| ≤ 5.99 for the same recordings. When analyzing the results in Chapter 4, we could argue that the N-point FFT method is quicker to react to changes in the breathing rate during the one-minute interval, and therefore show a more varying result. It can also be discussed how well a mean of a one-minute simulation represents the performance of the algorithms. However, with the possible future use of the algorithms in practice, it seems like the best way to quantify the results. Furthermore, it was not always easy to determine the true breathing rate, even for the recordings that made it into the data set, making them somewhat unreliable. Other trends noticeable in Table 4.13 is that the N-point FFT algorithm often scores a negative error margin, i.e., estimates a breathing rate lower than the true breathing rate. When comparing the two methods, one last point of discussion is that the Zero-Crossing algorithm is limited to a specific camera location, making it less versatile than the N-point FFT. Since the data set does not include many different angles, it is hard to argue that the N-point FFT algorithm can perform with other camera locations. However, we can not discard the possibility that it can perform under different circumstances.
This chapter presents the necessary conclusions that can be drawn from results and discussions from Chapter 4 and Chapter 5. Answers to the research questions asked in Section 1.3 are presented in Section 6.1 and possible future work or implementation are presented in Section 6.2.

6.1 Research Questions

This thesis started with four research questions in Section 1.3 formulated as:

- Can the respiratory frequency of cows be measured by computer vision and signal processing, and if so, how well does the measured frequency match the true respiratory frequency?
- Are there any performance deviations when using different types of image and signal processing algorithms, and if so, can the difference be quantified?
- Are there any performance deviations when using different types of optronic devices?
- How does the resolution of the input images correspond to the signal output and performance of the measured respiratory frequency?

Using computer vision and signal processing, the respiratory frequency of cows seems to be measurable with a reasonable margin of error. When studying the estimations presented in Section 4.3, both the methods proposed in Chapter 3 performs well on the majority of the data set in relation to the true respiratory frequency, with the delimitations presented in Section 1.4. The Breathing Move-
ment Zero-Crossing Counting algorithm calculates the optical flow with a Lucas-Kanade algorithm. It estimates a signal in the time domain that corresponds to the cow’s respiratory movements. This signal is passed through a bandpass filter and applied to a zero-crossings algorithm to find the signal’s fundamental frequency, which is the estimated breathing frequency of the cow. The N-point FFT algorithm estimates the breathing rate in the frequency domain by Fast Fourier Transform of motion vectors estimated with Horn-Schunk or Farnebäck’s algorithm and a Point-Mass inspired filter. A conclusion can be drawn, regarding the performance difference between the Breathing Movement Zero-Crossing Counting algorithm and the N-point FFT algorithm from Chapters 4 and 5. The performance level of the Zero-Crossing algorithm is higher and more consistent since the estimated breathing frequencies had margin errors $0.21 \leq |\epsilon| \leq 0.92$ for twelve of the thirteen recordings, where the cows were standing relatively still at the milking robot. However, unlike the N-point FFT algorithm, it is limited to a specific camera location above the cow, making it less versatile. The N-point FFT algorithm estimated the breathing frequencies with larger margin errors, $0.44 \leq |\epsilon| \leq 5.99$, for the same recordings. Still, it could make estimations on all sixteen recordings from the data set regardless of the camera location, with margin errors $|\epsilon| = 1.92$, $|\epsilon| = 7.00$, and $|\epsilon| = 10.88$ for the additional recordings, where the cows were laying down.

There can be no conclusion drawn regarding the performance deviations when using different optronics. The collected data set presented in Section 3.1 does not contain any recordings from two cameras positioned at the same location where a true respiratory rate could be determined. Regarding the resolution and how it corresponds to the signal output and performance of the estimated breathing rate, the conclusion drawn from this work is that increased resolution seems to give higher performance, at least for the N-point FFT algorithm. Both proposed algorithms show the same type of performance patterns in signal output for low resolution and high resolution, in relation to the simulations performed in Chapter 4 and discussed in Chapter 5.

### 6.2 Future Work

From the several test and discussions regarding execution time as part of the performance score, it would be of great interest to see an implementation of one or both algorithms on hardware to perform in real-time. It would also be interesting to see if the two algorithms could complement each other with a real-time implementation considering the performance differences.

A possible way to improve the work is to expand the data set for even more experimentation and tuning the parameter configuration. The problems expressed in Section 3.1 and discussions from Chapter 5 could assist a more comprehensive data collection. It would also allow getting data from outside the barn.

Furthermore, it would be of great interest to investigate the performance deviations when using different optronics. Since this work did not conclude that spe-
specific research question, a study based on potential deviations is highly relevant. This work has only used cows as subjects of estimation; it would be relevant to examine the performance of the algorithms on other animals or humans. Finally, an improvement of this work could be if one or both of the algorithms were introduced to machine learning or deep learning. For example, it could allow the opportunity to estimate multiple respiratory frequencies since the algorithm could distinguish between what is breathing and what is not.
Appendix
This appendix-chapter contains theory behind procedures used in Section 3 but not included in Section 2.

A.1 Zero Padding

An efficient way to get better FFT frequency resolution is to zero pad the time domain signal. The procedure is simply to extend the signal with zeros, which gives a longer time length, $T$, and since the waveform frequency resolution is defined by the equation:

$$
\Delta R = \frac{1}{T}
$$

(A.1)

it improves the resolution without losing information or adding any noise. The cost is an increase of computational complexity.

A.2 Fast Fourier Transform

Fast Fourier Transform (FFT) uses the definition of the Discrete Fourier Transform (DFT) of a $n$-long sample sequence of complex numbers, $x_0, ..., x_{N-1}$, to calculate the frequency domain output $X_k$:

$$
X_k = \sum_{n=0}^{N-1} x_n e^{-i\frac{2\pi kn}{N}}, \quad k = 0, ..., N - 1.
$$

(A.2)
Instead of using the definition directly, it is factorized into sparse factors, i.e., a matrix where most values are zero, to reduce the complexity of the computation.

Figure A.1: Example of N-point FFT algorithm structure

Figure A.1 is a modified version of Arthit Kosachunhanun's original image. The license to this image is found at https://creativecommons.org/licenses/by/3.0/deed.en
Bibliography


